Detecting Vulnerabilities in Smart Contract within Blockchain: A Review and Comparative Analysis of Key Approaches

Yoganand Kissoon
School of Digital Technologies,
Middlesex University Mauritius
Uniciti, Flic-en-Flac, Mauritius
YK299@live.mdx.ac.uk

Girish Bekaroo
School of Digital Technologies,
Middlesex University Mauritius,
Uniciti, Flic-en-Flac, Mauritius
g.bekaroo@mdx.ac.mu

***Abstract*— Blockchain technology was created with security in mind. However, in recent years, there has been various confirmed cases of breach, worth billions of dollars loss in Blockchain associated to smart contracts. In order to address this growing concern, it is crucial to investigate detection and mitigation of vulnerabilities in smart contract, and this paper critically reviews and analyses key approaches for detecting vulnerabilities in smart contract within Blockchain. In order to achieve the purpose of this paper, five key approaches, notably the application of OWASP Top 10, SCSVS, vulnerability detection tools, fuzz testing and the AI-driven approaches are critically reviewed and compared. As part of the comparison performed, a penetration testing quality model was applied to study six quality metrics, notably extensibility, maintainability, domain coverage, usability, availability and reliability. Results revealed limitations of the studied vulnerability detection approaches and findings are expected to help in decision making especially when selecting approaches to be used during security analysis and pen-testing.**

Keywords*— *Blockchain, Smart Contracts, Vulnerability Detection, Penetration Testing Methodologies, Security Analysis.

I. INTRODUCTION

During the previous decade, there has been rapid growth in usage of smart technologies and applications within domains such as smart healthcare and smart farming, among others [1]. Nevertheless, the use of smart technologies and applications has been hampered by security and privacy concerns due to the use of the publicly accessible network, notably, the Internet, for the transfer of data. Even though various security solutions and standards were developed for strengthening security of smart technologies and applications, these can potentially increase communication overheads and have limitations in terms of scalability, robustness and traceability, among others [1]. In order to address such issues, Blockchain technology can be a potential solution as this technology has also grown in prominence during recent years [2].

Blockchain is a record-keeping technology that has been designed with security as a key objective, such that it is practically impossible to hack the system or forge the data stored on it. In the same context, smart contracts are programs that are stored on a Blockchain and are executed automatically when some predetermined conditions are met. Nevertheless, there has recently been frequent outbreaks of smart contract security vulnerabilities and privacy issues that raised concerns and challenges to Blockchain [3, 4], given that applications of this technology are increasing within different fields. Such recent security issues even led to huge financial losses where for instance, the Dao security vulnerability in 2016 resulted in an economic loss of $50 million [5] and the security vulnerability of parity multi-signature wallet in 2017 resulted in loss of more than $150 million of ether [6].

Taking cognizance of the enormous growth in successful exploits of smart contract vulnerabilities in the past years and the significance in terms of monetary impact, it becomes crucial to investigate effective detection of vulnerabilities in smart contract. As such, this paper critically reviews and analyses key approaches for detecting vulnerabilities in smart contract within Blockchain. Findings presented in this paper is expected to provide different contributions to the the Blockchain and research communities in general. Firstly, the paper compiles and discusses the different vulnerability detection approaches that could be used during pen-testing and security analysis of smart contracts within Blockchain, which is relatively limited in published literature. Moreover, the findings following application of a chosen pen-testing quality model provides insights on different metrics pertaining to each vulnerability detection approach and this could help decision-making in the same context.

This paper is structured as follows: In the next section, a background on Blockchain smart contract technology is provided. Then, related works on vulnerabilities detection in smart contract are reviewed in Section III. Section IV describes the methodology used to identify and study key vulnerability detection approaches and Section V reviews the selected approaches. In the final sections, the existing vulnerability detection approaches are critically compared, before providing a conclusion related to the core of the related study area.

II. BLOCKCHAIN AND SMART CONTRACT TECHNOLOGY: A BACKGROUND

Blockchain is a method of storing data in such a way that it is challenging to alter, hack, or cheat. When Blockchain technology was first introduced by Haber and Stornetta in 1991, the initial idea was to invent a way to record documents and time stamp them in such a way that cannot be tampered. Data is collected in groups known as blocks, where each block holds a set amount of information which once filled, is encrypted, time stamped and chained together with the previously filled block. This activity thus creates a chain of blocks of information which gives the name Blockchain. The next step of the Blockchain technology is to distribute the information that has been recorded over number of participating nodes so that no one node has control over the information. Thus, the participating nodes also act as guardians of the information and can verify if the transaction being process is legitimate or not. The property of being a decentralized database which is dispersed across multiple participating nodes gives the name Distributed Ledger Technology (DLT) to Blockchain. This is important because

it gives birth to multiple other use case scenarios to be built upon this technique.

There are multiple properties of the DLT technology which makes this technology secure, as depicted in Fig. 1. The fact that the DLT is programmable means that codes can be added in the system to design specific use case scenarios for this technology. One of those programmable products is smart contracts [3]. An American scientist named Nick Szabo was the first to propose smart contracts in 1994. When it was invented, it was described as computerized operations that fulfil terms of a contract. The initial idea was to extend the capabilities of point of sales to the digital world. Szabo also mentions in his report that the technology could be used for other types of complex assets such as bonds. In other words, the technology of smart contracts could be used for sale or purchase of assets that are complex in terms and conditions [4].

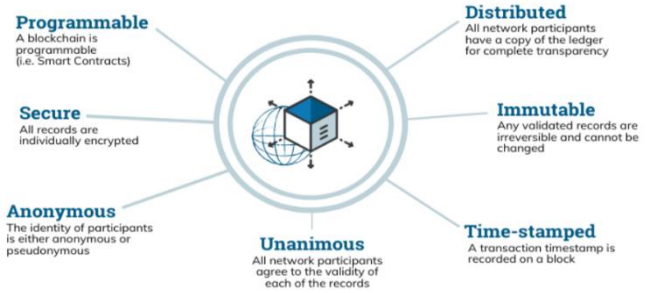

Fig. 1 Properties of Distributed Ledger Technology (DLT) [3]

In simple terms, a smart contract is a piece of computer codes that is programmed in a Blockchain, which cannot be altered, deleted, or hacked. The code is just terms and conditions that has been agreed between two parties and will self-execute without the need of any third party once a certain condition is met and the terms are fulfilled. Nick Szabo described the best metaphor for a smart contract could be a vending machine; thus, with a certain input of information by the buyer and a code of execution guaranteed by the computer system, a certain output is guaranteed [4]. The important properties that should be retained here are that the codes are written in such a way that they are self-executing, traceable, self-verifying, and temper proof [4]. The functioning of smart contracts is illustrated in Fig. 2 [5]. Smart contracts are popular in industries like property ownership, patents or intellectual property, banking and insurance, legal services, and crowdfunding organizations, among others.

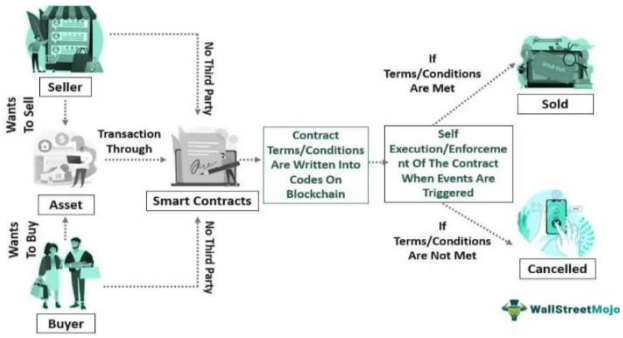

Fig. 2. Functioning of Smart Contracts [5]

As such, even though the smart contract technology is considered to be secure, it is not without vulnerabilities. For instance, the year 2021 amounts to an increase of more than 1300% and a mind boggling $2 Billion hacked from mainly smart contracts and Defi in the Blockchain network [6]. One of the most shocking and captivating stories of 2021 revolves around the successful exploit of a vulnerability on a smart contract in the poly network Blockchain which amounts to more than $600 Million [5]. Hence, security analysis and penetration testing of smart contract technology is essential to study and thus, the purpose of this paper becomes relevant to be addressed.

III. Related Works

The strong security nature of the baseline technology engulfing Blockchain, the complex architectures and the fairly young age of the technology means that there was not enough substantial need for research to be done in the field of security frameworks that would detect and remediate security or design flaws against smart contract in Blockchains. However, due to the issues discussed in the previous section and as highlighted in a previous study [8], there are crucial gaps that need to be tackled between the existing security frameworks and Blockchain in general, notably:

- New terminologies and definitions need to be outlined as compared to traditional OWASP used in the frameworks related to web and application security testing. Outdated framework is being used on other technologies that share certain traits with smart contract Blockchain.
- Multinational extension of Blockchain smart contract implied that the laws pertaining to smart contract across different countries has to be compiled as compliance benchmarks.
- Privacy protection has a high security score and is an essential integral part of security. However, in this case, nodes operate in a decentralized way and transaction are anonymous and thus, public nodes could be utilized in an unlawful way and not much could be done to detect/deter such attacks.

Another previous study [9] investigated the potential link between vulnerabilities detected in smart contract Blockchain and the exploitability potential of those flaws. The study revealed that out of the most common vulnerabilities available for smart contract in Blockchain, multiple tools and detection modes had to be used to detect those vulnerabilities, as depicted in Table I.

Table I. Mean Time for Scanning

Name	Vulnerabilities						Report month
	RE	UE	LE	TO	IO	UA	
Oyente	✓	✓		✓	✓		2016-10
ZEUS	✓	✓	✓	✓	✓		2018-02
Maian			✓			✓	2018-03
SmartCheck	✓	✓	✓		✓		2018-05
Securify	✓	✓	✓	✓		✓	2018-06
ContractFuzzer	✓	✓					2018-09
teEther						✓	2018-08
Vandal	✓	✓					2018-09
MadMax			✓		✓		2018-10

Within the same study [9], the list of common Smart Contract vulnerabilities was also provided, as described in Table II below.

TABLE II. LIST OF COMMON SMART CONTRACT VULNERABILITIES [9]

Vulnerability	Brief Description
Re-Entrancy (RE)	Caller is called back by malicious contract and funds are drained from the caller's account.
Unhandled Exceptions (UE)	Inconsistencies due to low level commands such as send continuing to execute even upon failure.
Locked Ether (LE)	ETH smart contracts like other smart contracts can bind funds in such a way that it is completely locked. If the smart contract that locked the funds are destroy the funds are permanently locked and cannot be transferred.
Transaction Order Dependency (TO)	Since in the same block, multiple transactions are possible, the smart contract will share the same property and can be updated multiple times, even by a malicious caller.
Integer Overflow (IO)	Programming language mistakes that can create a loop if thereby exploited by attacker by incrementing the iterations.
Unrestricted Action (UA)	The ability to set an owner without being allowed to.

Whilst different Smart Contract vulnerabilities adversely impact the security of such technology (as shown in Table II) and that there is not one analysis tool which can detect all the common types of vulnerabilities that exist on a particular smart contract Blockchain (Table I), an important question becomes important to investigate, notably, *what detection approach should be adopted*? The methodology provided in the next section describes the method used in order to answer this key question.

IV. METHODOLOGY

The primary source of research for this paper was carried out by screening research databases [14] and published penetration testing reports of real multination giants in the smart contract marketspace with the aim to gather details about pen-testing approaches for smart contract within Blockchain. The research databases filtered were IEEE Xplore and Google Scholar whereas for the published reports the website of the main actors in the market like Ethereum and Bitcoin were explored. The key terms used in the searching process include "Blockchain", "vulnerability detection", and "smart contract", among others. Following an initial pool of 11 results, filtering was conducted to assess relevance and meant the context of Blockchain. 5 such vulnerability detection approaches were identified and were eventually reviewed comprehensively by referring to the published resources. These approaches are discussed and critically compared in the next sections.

V. APPROACHES FOR DETECTING VULNERABILITIES IN SMART CONTRACT WITHIN BLOCKCHAIN

Using the methodology defined in the previous section, the selected vulnerability detection approaches are discussed as follows:

A. OWASP Top 10

According to previous studies [8, 10], the Open Web Application Security Project (OWASP) Top 10 was found to map well to the baseline architecture of smart contract and Blockchain. The OWASP Top 10 is a list of the 10 most severe security issues as defined and regularly updated by the OWASP community. Though the project is limited to those 10 main categories, the OWASP Top 10 also provides information about industry vulnerabilities and the integral framework to test them. OWASP has been applied for penetration testing of Bitcoin smart contract, where different vulnerabilities were revealed, as shown in Fig. 3. These vulnerabilities were eventually analysed and appropriate recommendations were made towards enhancing security [11].

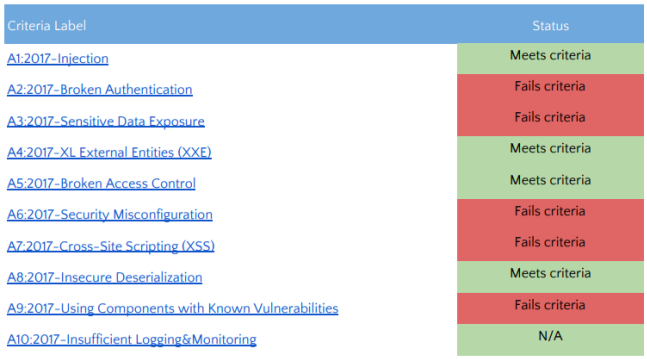

Fig. 3. Application of OWASP to Bitcoin Smart Contract [11]

B. SCSVS

Smart Contract Security Verification Standard (SCSVS) is regarded as the next evolutionary phase in the effectiveness of the penetration testing activity for smart contracts [12]. While OWASP is regarded as an effective approach, it has been designed with web applications in mind. However, decentralized applications and smart contract Blockchain have a slightly different trait, as illustrated in Fig. 4 . This fact implies that OWASP as penetration testing framework is likely to have components that are relevant to smart contracts, as also highlighted in previous research [11, 12]:

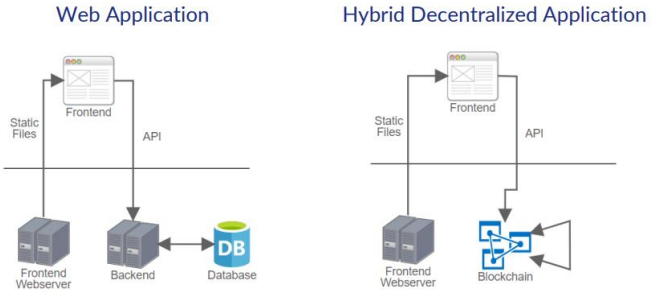

Fig. 4. Architectural Difference Web App vs Smart Contracts [12]

Based on the OWASP Application Security Verification Standard (ASVS), the SCSVS (v1.2) consists of 14-part checklist developed with the aim to standardize security of smart contracts [13]. This list can be used by key stakeholders of smart contract including developers, architects, security reviewers as well as vendors and provides useful guidance in order to prevent key security issues at every stage of the development cycle of smart contracts.

C. Using Vulnerability Detection Tools

A previous study proposed a detection framework based on a list of most common vulnerabilities for smart contracts as outlined in Table 2 and using a list of automated software that

can detect those vulnerabilities [14]. The techniques used involve:

- code translation: recompiling or decoding the code used into another form so that it can be interpreted in a way that permits the detection of vulnerability
- static analysis: analysis of smart contract codes without execution;
- dynamic analysis: executing the code in an environment where detection of vulnerabilities is possible.

According to the same study, the right combination of detection tools can potentially generate an effective result, also based on the results of the comparative analysis of the tools shown in Table III. The limitations of these techniques are however based on not relying on one particular tool or technique for penetration testing, it is rather a combination of tools that will provide a successful result.

TABLE III. RESULTS OF COMPARATIVE ANALYSIS OF VULNERABILITY TOOLS FOR SMART CONTRACTS [14]

Tool	Method						Supported Vulnerability Type						
	Disassembly	*CFG*	*Pattern matching*	*Symbolic execution*	*Result validation*	*Fuzzing*	*IO*	*PVM*	*EHE*	*TOD*	*PRN*	*RE*	*FT*	*FF*
Oyente	√	√		√					√	√	√	√		
Securify	√		√					√	√	√			√	
Mythril	√	√		√			√	√	√		√	√	√	
Manticore	√			√			√	√	√			√		
teEther	√	√		√	√			√						
MAIAN	√	√		√	√			√					√	
ContractFuzzer						√			√		√	√	√	
EVulHunter	√	√	√											√
EOSafe	√	√		√				√			√			√

D. *Fuzz Testing*

Previous studies have developed and adopted fuzz testing as baseline proposal for vulnerability testing for smart contracts [15, 16]. Fuzz testing, also known fuzzing, is a black-box software testing technique used to find bugs in an automated way through the injection of malformed/semi-malformed data [17]. A previous study [15] outlined an architecture for fuzz testing for Smart Contracts as shown in Fig. 5:

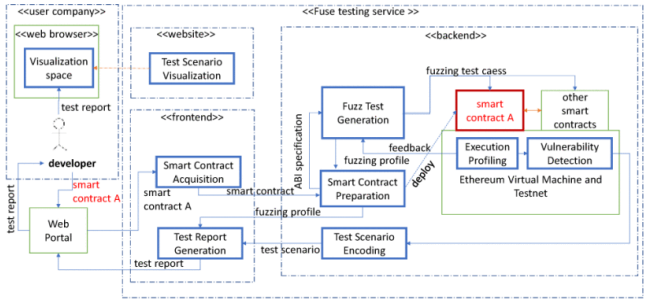

Fig. 5. Fuzz Testing Architecture for Smart Contracts [15]

In this architecture, the smart contracts that were tested generated a high quality result, meaning that the number of false positives were minimal and a high number of different vulnerabilites class for smart contracts were detected as summarized in Table 5 [16]. The vulnerabilities detected were classified into different categories, but the true strength of this approach lies in the accuracy of the detection, where little false positives were identified. This implies that the overall efficiency of this technique is commendable.

TABLE IV. FUZZ DETECTIONS FOR SMART CONTRACTS [16]

Vulnerability Class	# of Detected Vulnerabilities	True positive rate
Gasless Send	138	1.000
Exception Disorder	36	1.000
Reentrancy	14	1.000
Timestamp Dependency	152	0.960
Block Number Dependency	82	0.963
Freezing Ether	30	1.000
Dangerous Delegatecall	7	1.000

E. *AI-Driven Approach*

Previous studies [18, 19] proposed the approaches of machine learning, artificial intelligence, and deep learning integrated within vulnerability detection for smart contracts. In these studies, the fundamental approach involves building a system that can automatically evolve into more effectively detecting vulnerabilities in smart contracts. Such a proposed architecture for smart contract vulnerability detection using the mentioned approaches is illustrated in Fig. 6.

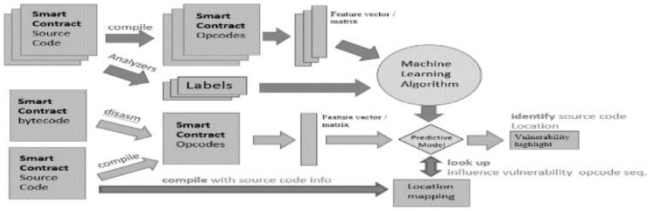

Fig. 6. AI architecture for Smart Contract Vulnerability Detection [19]

This architecture in Fig. 6 consists of using machine learning algorithm into training an artificial intelligence (AI) model that can understand what smart contracts are and the vulnerabilities associated to them. In the proposed model in a previous study [18], AI was used to learn detection of 13 types of smart contract vulnerabilities using Oyente and Remix detectors. Once the model was trained, it was used to automatically detect vulnerabilities in smart contracts with a high level of accuracy, as illustrated in Table V. In the same study, different AI learning models were used such as logistic regression, SVM linear, SVM kernel, K-Nearest Neighbor, Decision Tree, Randon Forest and Gradient Boosting. Among these algorithms, it was found that the Logistic Regression model provides a high level of accuracy and precision and that the final score is above the other models.

TABLE V. AI DETECTION RESULT FOR SMART CONTRACT VULNERABILITIES [18]

Feature Extraction Methods	*Models*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
n-gram + tf-idf	**Logistic Regression**	**97.3%**	**92.9%**	**88.2%**	**90.4%**
	SVM Linear	94.8%	88.8%	80.3%	84.1%
	SVM Kernel	86.7%	63.9%	54.9%	57.2%
	K-Nearest Neighbor	93.5%	85.9%	70.2%	76.3%
	Decision Tree	95.7%	87.0%	82.0%	84.4%
	Random Forest	95.8%	95.4%	67.3%	75.1%
	Gradient Boosting	96.6%	91.0%	85.3%	87.7%

VI. Critical Analysis

In order to critically analyse the selected vulnerability detection approaches, the penetration testing quality model described in ISO/IEC 25010:2013 was adapted and used. The same adapted model was used in previous published research [20] related to comparative analysis of penetration testing frameworks and is thus relevant to this study. An illustration of the model is provided in Fig. 7.

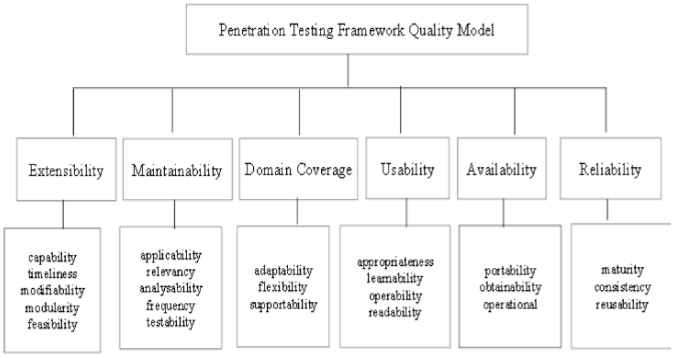

Fig. 7. Penetration Testing Quality Model [20]

The framework has different qulality metrics, notably:

- Extensibility: involves assessing how easy it is to modify or extend the approach in order to add new components.
- Maintainability: relates to assessing how easy it is to maintain the approach.
- Domain coverage: relates to the scope of the approach whereby evaluating if the approach covers sufficient areas within its context.
- Usability: entails assessing how easy it is to use and apply the overall approach.
- Availability: involves evaluating if the approach is available for use whenever needed.
- Reliability: entails measuring if the approach is reliable sufficiently such that it can sustain different conditions such as different application scenarios and environments.

The above quality metrics were applied to the selected approaches reviewed in this study using similar method involved in the previous study [20], whereby comprehensively referring to published resources pertaining to each approach. Findings are presented in Table VI and results showed that none of the vulnerability detection approaches meet all the quality metrics. To start with, although OWASP is popular for web systems, it does not cover all areas in relation to smart contracts, as shown in Fig. 3. As such, it is not fully reliable for complete pen-testing of smart-contracts. The same findings was noted for the adoption of the SCSVS approach. As such, vulnerability detection approaches like OWASP or SCSVS are effective in detecting vulnerabilities in smart contracts and are normally applied at an interval, notably when the company owning the smart contract decides to invest in a penetration testing activity for its Blockchain. As such, the key limitation is that smart contracts are constantly being generated and if the interval is lengthy, vulnerabilities could be exploited by attackers.

On the other hand, even though the use of a combination of vulnerability detection tools was found to meet most of the criteria including domain coverage and reliability, some of the tools can be challenging to acquire due to costs or licenses involved. Furthermore, due to the nature of inputs and data involved in the fuzz testing approach, maintainability and usability are key contstraints noted. The Fuzz testing models, or dynamic fuzz models would be more efficient for smart contracts because these models detect vulnerabilities constantly at entry point. However, fuzz testing also has its limitations since vulnerabilities constantly evolve. This implies that new vulnerabilities need to be tested with newer kinds of inputs, thus impacting maintainability of such approach.

Finally, AI-Driven approach, was found to meet most quality metrics besides availability as implemented algorithms are mostly propriatery (part of research projects or publications) or implemented within tools, that could be challenging to acquire. Overall, each approach outlined in this study have strengths and weaknesses in specific subdomains. This is also highly correlated with the fact that the smart contract Blockchain technology itself is growing in maturity.

Even though the comparative analysis through the use of different quality metrics provided insightful findings regarding approaches used to detect vulnerabilities in smart contracts within Blockchain, different limitations also undermine the results provided in this study. For instance, findings were based on published information and could be better validated through practical application of different approaches in order to derive more critical insights regarding each approach.

VII. Conclusion

In this paper, different factual realisation points were noted. The fact that Smart Contracts within Blockchain are undeniably vulnerable to multiple types of security issues leads to the need for penetration testing and security analysis

TABLE VI. Analysis of Vulnerability Detection Approaches using Pen-Testing Quality Model

Pen-Testing Approach	Quality Metric					
	Extensibility	**Maintainability**	**Domain Coverage**	**Usability**	**Availability**	**Reliability**
OWASP Top 10	✓	✓	×	✓	✓	×
SCSVS	✓	✓	×	✓	✓	×
Using Vulnerability Detection Tools	✓	✓	✓	✓	×	✓
Fuzz Testing	✓	×	✓	×	✓	✓
AI-Driven Approach	✓	✓	✓	✓	×	✓

in order to detect vulnerabilities in smart contracts in a timely manner. Five key vulnerability detection approaches were investigated through the application of an adapted penetration testing quality model described in ISO/IEC 25010:2013 to study six quality metrics, notably extensibility, maintainability, domain coverage, usability, availability and reliability. Results revealed that all the approaches have their limitations. For instance, the application OWASP Top 10 and SCSVS were limited in their domain coverage as both approaches do not fully cover all areas of pen-testing for smart contract. As such, their complete reliability are also questionable for the context of vulnerability detection in smart contracts. Furthermore, even though using vulnerability detection tools and AI-driven approaches can help to detect various classes of vulnerabilities, it is not easy to acquire some of them. In addition, the usability of fuzz testing is limited due to the characteristics of data and inputs needed in the process. As such, the best approach would be a combination of approaches whereby involving AI with reinforcement learning that constantly learns following pen-testing instances in order to produce strengthened models that can be used to detect vulnerabilities in Smart Contract source codes in a predictive manner. As future works, the limitations identified in this study can be further investigated whereby practically applying different vulnerability detection approaches in order to derive further insights.

REFERENCES

[1] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar and M. Alazab, "Blockchain for industry 4.0: A comprehensive review," *IEEE Access*, vol. 8, pp. 79764-79800, 2020.

[2] D. Berdik, S. Otoum, N. Schmidt, D. Porter and Y. Jararweh, "A survey on blockchain for information systems management and security," *Information Processing & Management*, vol. 58, no. 1, p. 102397, 2021.

[3] X. Tang, K. Zhou, J. Cheng, H. Li and Y. Yuan, "The Vulnerabilities in Smart Contracts: A Survey. In International Conference on Artificial Intelligence and Security," Cham, 2021.

[4] S. Sayeed, H. Marco-Gisbert and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24416-24427, 2020.

[5] I. Sergey and A. Hobor, "A concurrent perspective on smart contracts," in *International Conference on Financial Cryptography and Data Security*, Cham, 2017.

[6] T. Bocek and B. Stiller, "Smart contracts – blockchains in the wings.," in *Digital Marketplaces Unleashed*, Heidelberg, Springer, 2018, p. 169–184.

[7] Euromoney Learning, "What is blockchain?," Euromoney, 2022. [Online]. Available: https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain. [Accessed 14 Jan 2022].

[8] J. Frankenfield, "Smart Contracts," Investopedia, 2022. [Online]. Available: https://www.investopedia.com/terms/s/smart-contracts.asp. [Accessed 2 Feb 2022].

[9] D. Vaidya, "Smart Contracts," WallStreetMojo, 2022. [Online]. Available: https://www.wallstreetmojo.com/smart-contracts/ . [Accessed 10 Feb 2022].

[10] R. Behnke, "The 10 Biggest Defi hacks of 2021," Halborn, 2022. [Online]. Available: https://halborn.com/the-10-biggest-defi-hacks-of-2021-a-recap/. [Accessed 3 Feb 2022].

[11] G. Chavez-Dreyfuss and M. Price, "Explainer: How hackers stole and returned $600 mln in tokens from Poly Network," Reuters, 2021. [Online]. Available: https://www.reuters.com/technology/how-hackers-stole-613-million-crypto-tokens-poly-network-2021-08-12/. [Accessed 14 Feb 2022].

[12] A. Bhardwaj, S. Shah, A. Shankar, M. Alazab, M. Kumar and T. Gadekallu, "Penetration testing framework for smart contract blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2635-2650, 2021.

[13] D. Perez and B. Livshits, "Smart contract vulnerabilities: Vulnerable does not imply exploited," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[14] A. Zakari, A. Lawan and G. Bekaroo, "Towards improving the security of low-interaction honeypots: Insights from a comparative analysis," in *International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering*, Cham, 2016.

[15] H. Poston, "Mapping the OWASP top ten to blockchain," in *Procedia Computer Science*, 2020.

[16] Under Defence, "Penetration Testing Report for Bitcoin Exchange Company," 2018.

[17] D. Rusinek, "Secure Smart Contracts Development using SCSVS," OWASP, 2022.

[18] D. Rusinek and P. Kuryłowicz, "Smart Contract Security Verification Standard," GitHub, 2022. [Online]. Available: https://securing.github.io/SCSVS/. [Accessed 10 Apr 2022].

[19] J. Xu, F. Dang, X. Ding and M. Zhou, "A Survey on Vulnerability Detection Tools of Smart Contract Bytecode," in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 2020.

[20] X. Mei, I. Ashraf, B. Jiang and W. Chan, "A fuzz testing service for assuring smart contracts," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2019.

[21] W. Chan and B. Jiang, "Fuse: An architecture for smart contract fuzz testing service," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018.

[22] OWASP, "Fuzzing," OWASP, 2022. [Online]. Available: https://owasp.org/www-community/Fuzzing. [Accessed 13 Feb 2022].

[23] J. Liao, T. Tsai, C. He and C. Tien, "Soliaudit: smart contract vulnerability assessment based on machine learning and fuzz testing," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, 2019.

[24] F. Mi, Z. Wang, C. Zhao, J. Guo, F. Ahmed and L. Khan, "VSCL: Automating Vulnerability Detection in Smart Contracts with Deep Learning," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021.

[25] A. Shanley and M. Johnstone, "Selection of penetration testing methodologies: A comparison and evaluation," in *Australian Information Security Management Conference*, 2015.