World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:16, No:5, 2022

Security Strengths and Weaknesses of Blockchain Smart Contract System: A Survey

Malaw Ndiaye, Karim Konate

Abstract—Smart contracts are computer protocols that facilitate, verify, and execute the negotiation or execution of a contract, or that render a contractual term unnecessary. Blockchain and smart contracts can be used to facilitate almost any financial transaction. Thanks to these smart contracts, the settlement of dividends and coupons could be automated. Smart contracts have become lucrative and profitable targets for attackers because they can hold a great amount of money. Smart contracts, although widely used in blockchain technology, are far from perfect due to security concerns. Although a series of attacks are listed, there is a lack of discussions and proposals on improving security. This survey takes stock of smart contract security from a more comprehensive perspective by correlating the level of vulnerability and systematic review of security levels in smart contracts.

Keywords—Blockchain, bitcoin, smart Contract, criminal smart contract, security.

I. Introduction

A Smart contract is a collection of promises in digital form and protocols within which the parties execute those promises [1], [2]. It may enforce specific activities such as addressing financial fraud, e-voting, bug bounty and the blockchain-Internet of things (IoT) combination. Moreover, it can be applied to cloud computing to enforce payments. However, smart contracts may cause significant damage if they are targeted by criminals [3], [4]. The importance of smart contract integration of blockchain technology become a focus area to develop because it gives peer to peer transaction and database can be maintained publicly in a secure way, in a trustful environment [5].

Smart contracts, although widely used, are far from perfect because of potential security issues. Although there are some recent studies on smart contracts security [6], [7], [8], [9], none of them perform a systematic examination on the strength and weakness of security on smart contracts. From security programming perspective, their is to analyze the security vulnerabilities of Ethereum smart contracts, and provide a taxonomy of common programming pitfalls that may lead to vulnerabilities. However a series of related attacks on smart contracts are listed, there is a lack of discussion on security. This survey focuses on the security of smart contracts from more comprehensive perspectives. The main contributions of this paper are as follows:

- We perform a vulnerability analysis to show the security weaknesses of smart contracts by giving a taxonomy of vulnerabilities (vulnerabilities related to platforms, programming languages and virtual machines)
- To our knowledge, we perform the first systematic review of smart contracts security levels.
- Then, we propose solutions to completely solve the problem of the immutability of smart contracts in order to offer them the possibility of being corrected even once deployed.

Malaw Ndiaye and Karim Konate are with Cheikh Anta Diop University, Senegal (e-mail: malaw.ndiaye@ucad.edu.sn, karim.konate@ucad.edu.sn).

The last part of this paper is organized as follows: Section II introduces the smart contracts main technologies used in blockchain systems. Section III systematically examines the security of smart contracts by highlighting security flaws and surveys the smart contracts security level on blockchain systems. After, we discuss a few future directions in Section IV. Finally, we conclude the paper.

II. Overview of Smart Contracts Technologies

In this section, we give an overview of smart contracts. First, we make a brief introduction to smart contracts, and then present the Smart Contracts technology.

A. Overview

1) What are Smart Contracts?: Smart contracts can be defined as custom logic and code deployed and run on blockchain platform. Smart contracts are digitized and codified transaction rules between accounts [10]. They facilitate the transfer of digital assets between accounts as an atomic transaction. Smart contracts can store data, and that can be used to record information, facts, balances, and any other information needed to implement real-world contract logic [10], [11], [12]. Smart contracts can be used to automate many different processes in areas such as insurance, real estate, supply chains, data management, identity management and voting. With the growth of the IoT, smart contracts could also play a central role in machine interactions [13], [14], [15].

2) Why Smart Contracts?: Although variations of smart contracts existed in the 1990s, lack of the requisite technology prevented widespread implementation. Prior to blockchain, smart contracts were computer programs which facilitated negotiation, verified and enforced performance on a centralized server [16]. Financial institutions used a form of pre-blockchain smart contracts when they eased bookkeeping transactions and option contracts by implementing computer code. However, general uncertainty and concern from users, combined with issues of identity and transaction verification ultimately hindered the use of smart contracts. Blockchain technology confronted these obstructions and has since molded the use of smart contracts [11], [16]. Once developed, blockchain streamlined the use of smart contracts, serving

as its technological framework and providing security and accuracy:

- First, blockchains and other distributed ledgers can maintain an immutable record of data and effectively mitigate single points of failure.
- Second, since smart contracts inherit the encryption pseudonymity of blockchains, even with the code and data accessible to everybody, smart contracts are able to prevent unwanted monitoring and tracking [11].
- Third, thanks to the flexibility of programming languages, smart contracts also have a good interoperability among multiple instances and are able to tackle modification better [11].

B. Smart Contracts Technologies

The importance of smart contracts integration of blockchain technology become a focus area to develop because it gives peer to peer transaction and database can be maintained publicly in a secure way and a trustful environment. Fig.1 shows the structure of a smart contract. It includes a set of executable functions and state variables. The contract code is executed on each node participating in the network as part of the verification of new blocks [17]. Fig. 2 shows the smart contract mechanisms.

There are deterministic and non-deterministic smart contracts. A deterministic smart contract does not need any information from an external party (outside of the blockchain). A non-deterministic smart contract depends on oracles or data flows from an external party. [12]. So, smart contract technology is based on three things: the platform, the programming language and the execution environment (the virtual machine).

- Smart Contract Platform: is a group of technologies that are used as a base upon which smart contracts, other processes or technologies are developed [18] [19]. However, each also has its specific characteristics, opportunities, and challenges, which can quickly change over time. Table I illustrates a study of the top 5 platforms and their characteristics [20], [21], [22].
- Smart Contract programming language: programming language is a formal language, which comprises a set of instructions that produce various kinds of output [23], [24].
- Smart Contract runtime environment: A runtime environment is the execution environment provided to an application or software by the operating system [25].

Since we have talked a bit about the informal framework for evaluating smart contracts platforms and their technologies, we can look at some facts and figures that might be sobering on the competitive landscape. The first most direct way to see which networks have the greatest developer activity, and are therefore used and worked on, is by comparing their activity (Table I). Finally, based on indicators such as market capitalization, development activity, daily active addresses and the number of transactions on chain (Figs. 3-5), we can not say that Ethereum is better or not. But we can conclude that today the most used smart contracts platform in this regard is Ethereum [26].

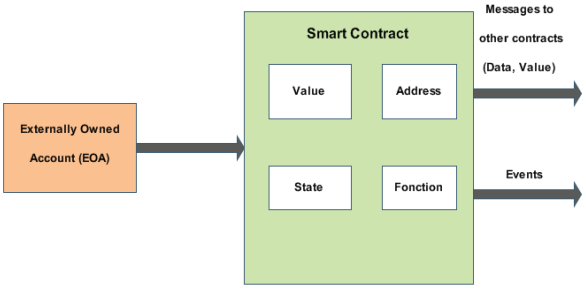

Fig. 1 Smart contract structure [5]

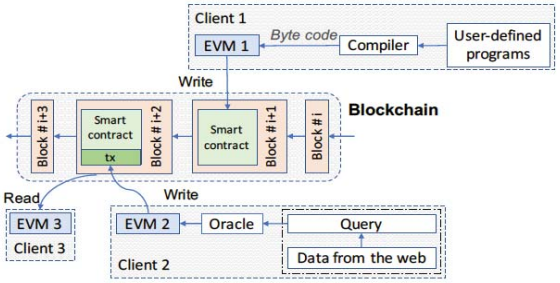

Fig. 2 A prototype of smart contracts mechanism

III. Smart Contract Security

This section first covers the related work on research papers and other articles that analyze/identify and provide an overview of security vulnerabilities in smart contracts. Moreover, each vulnerability is explained, in order to have a better understanding of the attacks and incidents that will be mentioned. This allows us to measure the risks that exist on smart contracts in order to propose solutions for risk

TABLE I
Top 5 of Most Popular Smart Contracts Platform [27]

Platforms	Programming languages	Runtime Environment
Ethereum	Solidity	EVM (Ethereum)
EOS	C/C++ (compiles to WASM)	WVM(web assembly virtual machine)
TRON	Solidity	TRON Virtual Machine (TVM)
Cardano	Plutus	IELE VM/KEVM
Waves	Ride	JRE

Fig. 3 Smart contracts market cap

TABLE II
SMART CONTRACTS PLATFORM STATISTICS [28], [29], [30], [31]

Smart Contracts	Market cap	Development Activity	Daily Active Addresses
Ethereum	$19 029 890 302	143,93	238981,03
EOS	$271 533 407	83,53	137589,07
TRON	$1 006 896 557	31,80	0
Cardano	$211 023 781	353,87	No data
Trezo	$631 547 290	0	No data
NEO	$674 880 181	17,83	No data
NEM	$361 491 491	0	No data
Ontology	$558 061 960	8.83	No data
VeChain	$294 975 765	1,77	0
Qtum	$165 701 334	10,27	No data
Algorand	$754 758628	26,40	No data
Waves	$81 537 817	106,67	No data

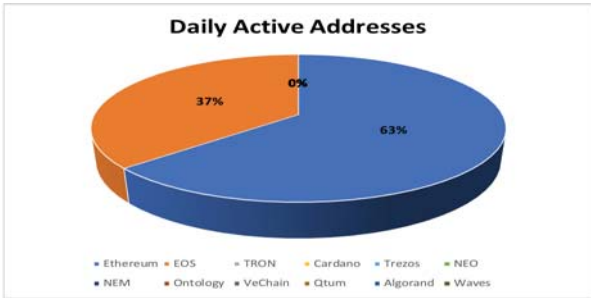

Fig. 4 Daily active addresses

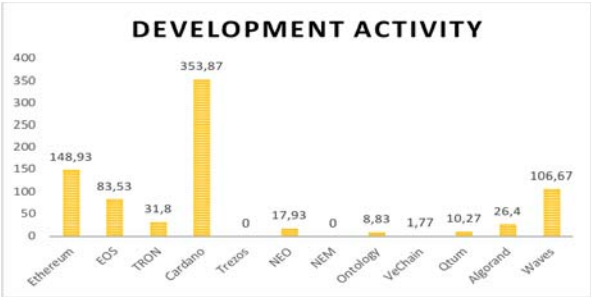

Fig. 5 Development activity

management. Secondly, our analysis focuses on security level evoked in related works and we try to show their limits because we know that, in safety, the level zero risk does not exist.

A. Smart Contracts' Vulnerabilities

A vulnerability or flaw is a weakness in a computer system that allows an attacker to compromise the integrity of that system, that is, its normal operation, the confidentiality, or the integrity of the data that it contains [32]. These vulnerabilities are the result of weaknesses in the design, implementation, or use of a hardware or software component of the system, but they are often software anomalies related to programming errors or bad practices.

Using smart contracts necessarily involves digitising the entirety of the transaction between the parties, which arguably exposes them to greater risk of sensitive information being compromised [33]. The particularity of a smart contract is to be immutable once deployed. If the correction of smart contracts is therefore a major lock, it is important to note that most attacks were caused by bugs or vulnerabilities of the execution platform [34], the programming language and the virtual machine [23], [6]. To go beyond a proof of concept, it will be appropriate to consider the development of smart contracts with precautions similar to those we take for the development of critical codes. A platform for critical applications would, from a technical point of view, need a verifiable input language as well as a proven compiler chain and a secure runtime environment. Aware of the security challenges and the problems encountered on smart contracts, we will address the following points: Platforms vulnerabilities, programming languages vulnerabilities and virtual machines vulnerabilities.

1) Platforms' vulnerabilities: The objective is to make an empirical evaluation of vulnerabilities on smart contract platforms (Table III). The conclusion of Section II-B shows clearly that Ethereum is the most used platform. Despite its popularity, surveys in [24], [7][21] show that Ethereum is the most vulnerable platform (Fig. 6) with its language and execution environment(EVM).

Ethereum two-version platform: A significant attack on one of the biggest smart contracts ever deployed, the DAO, led to notable damage to the idea of immutability and Ethereum in general. Since, The DAO attack had an enormous negative impact on Ethereum . However, simply said, a hacker was able to steal $80 million worth of Ether [35] from a complex smart contract. As a result, the community was split in now two platforms, the new Ethereum branch and the old Ethereum branch now called Ethereum Classic. It is worth mentioning that these platforms now, function as completely two separated platforms with their own cryptocurrency and community. Whereas, Ethereum (the new branch), except for the disadvantage of not staying true to the notion of immutability, everything else, such as the exponential growth and the constant updates and modifications, are considered to be advantageous [7].

2) Programming Languages' vulnerabilities: After having reviewed several articles on smart contracts programming languages, we realize that even the most experienced developers are not immune to vulnerabilities. For example in [23], in November 2017, a developer whilst fixing a bug that let attackers steal 32 million USD from a few multi-signature wallets accidentally left a second bug in the

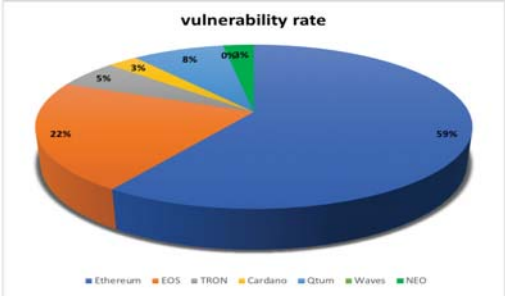

Fig. 6 Vulnerability rate in all implemented contracts

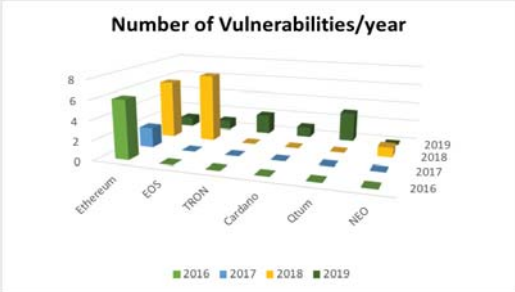

Fig. 7 Number of vulnerabilities per year

system that allowed one user to become the sole owner of every single multi-signature wallet. The above incidents show that even the most experienced developers can leave behind security vulnerabilities and bugs that are exploitable and failure prone. It is important to differentiate vulnerability defined in Section III and bug. Table IV presents a portrait of the vulnerabilities and bug related to smart contracts programming languages.

Despite its high degree of vulnerability, Solidity offers more advantages than other smart contract programming languages. Among these advantages, we can mention its compatibility with other platforms such as Quorum, Wanchain, Aeternity, Rootstock (RSK), Qtum, DFINITY etc. [27], [36]. We also can mention the usability of the language which is expressed with a rate of 80% [23].

3) Virtual Machines' Vulnerabilities: Virtual Machine is the run-time environment for smart contracts and its vulnerabilities may lead to serious problems to the platform ecosystem [43]. With lots of techniques being continuously developed for the validation of smart contracts, the testing of Virtual Machine remains challenging because of the special test input format [44]. Despite the scarcity of virtual machine analysis tools, some vulnerabilities have been discovered on the Ethereum virtual machine (EVM) and Webassembly virtual machine (WAVM). Table V shows the list of vulnerabilities for EVM and WAVM.

B. Smart Contracts' Security Level

Computer security is the set of technical, organizational, legal and human means necessary for the establishment of means to prevent the unauthorized use, modification or misuse of the information system [48]. Integrity is defined as the assurance that information is not altered or modified except by properly authorized individuals [49]. It is in the wake of strengthening the security of transactions in the chain that smart contracts technology has been put in place. These smart contracts are themselves victims of many vulnerabilities. Unfortunately in many articles, the security of smart contracts has not received much attention, although several anecdotal incidents of smart contracts malfunctioning have recently been reported, including contracts that do not execute as expected [50]. Outside the consensus mechanisms, the analysis of the previous works [51], [52], [53], [54], [55] allows us to highlight that different security components make it possible to ensure the resistance to attacks of smart contracts. These components are grouped into two categories, static checks and dynamic checks. The static verification is interested in checking the code before it is executed by the system. The dynamic verification ensures safety during the execution of the application [56]. In this subsection we will discuss the security offered by formal verification, vulnerability detection and access control.

1) Formal verification methods: Formal methods are special types of mathematical techniques for the specification, development and verification of software and hardware systems [57]. Formal methods are used in the design of software and hardware. This use is motivated by the hope that performing an appropriate mathematical analysis can contribute to the reliability and robustness of a design [58]. A software test is made up of validation and verification. Validation consists of determining whether the product meets the requirements and verification consists of verifying whether or not the product meets specifications [59]. There are several different methods for formal verification such as structure checking, type checking, model checking, theorem checking, etc., each with their own strengths and weaknesses. The most used in the field of smart contracts are theorem proving and model checking [56], [14], [54]. So Table VI shows a comparative study of some formal verification algorithms proposed in [50], [51], [52], [60], [53], [61], [54].

2) Vulnerabilities' Detection: The significant or even catastrophic impact of vulnerabilities or flaws or bugs in a system has led users to increase their efforts for the definition of methods and the development of tools for the detection and elimination of these types of programming errors from the early stages of software design. Among the techniques and tools relatively used in the field of blockchain, we can mention formal methods, verification and validation as well as static and dynamic code analyzers [26].

Given the immutability of smart contracts, the detection on intrusion via tools will allow developers to correct flaws in future smart contracts. Vulnerabilities detection tools cited in [67], [68], [69], [55], [44], [8], [7] have either static analyzer characteristics or dynamic analyzer characteristics. Certainly, vulnerability detection is a very important element in the security but detection must be accompanied by correction. Table VIII provides an overview of tools and methods of analysis.

3) Access Control Mechanism: Access control is a way to protect security by detecting, preventing unauthorized

TABLE III
TAXONOMY OF VULNERABILITIES ON SMART CONTRACTS PLATFORMS [26], [7], [37], [38], [39]

Platform name	Vulnerability name	Vulnerability Type(s)	Published date
Ethereum	Allocation of Resources Without Limits or Throttling	DoS	2018
	NULL Pointer Dereference	DoS	2018
	Deserialization of Untrusted Data	DoS	2019
	Out-of-bounds Read	DoS+Inf	2018
	Out-of-bounds Read	DoS	2018
	Incorrect Authorization	Bypass	2018
	Re-entrancy problem	The DAO attack	2016
	Transaction ordering	No Data	No data
	Block timestamp dependency	No Data	
	Exception handling	The DAO attack, Integer Over/Under flow attack, King of Ether Throne attack	2016
	Call stack depth limitation	no data	no data
	Integer overflow/underflow	Integer Over/Under flow attack	No data
	Unchecked and failed send	The DAO attack	2016
	Destroyable / suicidal contract	Parity Multisig Wallet attack	No data
	Unsecured balance	Integer Over/The DAO attack, Parity Multisig Wallet attack	2016
	Misuse of ORIGIN	no data	No data
	No restricted write	Parity Multisig wallet attack	2017
	No restricted transfer	”The DAO attack, Parity Multisig wallet attack”	2016
	Non-validated arguments	IInteger Over/Under flow attack	2018
	Greedy contract	Parity Multisig Wallet attack	2017
	Prodigal contract	The DAO attack	2016
	Gas overspent	Contract code execution consumes more gas unnecessarily	No data
EOS	Improper Restriction of Operations within the Bounds of a Memory Buffer	Overflow	2018
	False Top-Up	No data	2019
	Numerical Overflow	Numerical Overflow	2018
	Authorization Check	No data	2018
	Apply Check	No data	2018
	Transfer Error Prompt	No data	2018
	Random Number Practice	No data	2018
	Rollback Attack	No data	2018
TRON	Urgent/11	IP stack	2019
	DOS attack by consuming all CPU and using all available memory	DoS	2019
Cardano	fake stake	PoS	2018
Qtum	qtum through 0.16	Remote DoS	2019
	Block Flood Attack	DoS	2019
	’Fake Stake’ Attack	PoSV3	2019
Waves	No data	No data	No data
NEO	Runtime Serialize Calls DoS	DoS	2018

access and preventing unauthorized access and allowing authorized access in an automated system. An access control mechanism includes hardware or software functionalities, operating procedures, management procedures and various combinations of these functionalities [70]. There are two different types of access control mechanisms: user-based and host-based. The access control mechanism is based on two elements: authentication and authorization [64], [71]. Authentication is the process of verifying the identity of a user by obtaining some kind of credentials and using these credentials to verify the identity of the user. After the authentication, the authorization determines the permission to access on the objects. The access control mechanisms used by smart contracts [63], [62], [64], [66], [65] are based on mathematical and cryptographic principles to ensure confidentiality and integrity of the system. Table VII shows the analysis done on some papers on access control mechanisms.

IV. DISCUSSION AND OPEN ISSUES

Now that we have talked about smart contracts, their strengths and security weaknesses, we can discuss some facts and figures that could guide our reflexion about the major challenges of blockchain technology, especially smart contracts [73].The statistics in Section III-A1 show that Ethereum is one of the most used smart contract platforms. This thesis is demonstrated by its market capitalization (79%), by the development activity and the daily activity address rate. Ethereum popularity can be explained by its compatibility with other smart contracts programming languages and this domination gives it a higher average vulnerability rate (69%).The vulnerability of smart contracts operating on three levels (platform, virtual machine and programming language) is a weak link on the security of the blockchain technology. We are out of 21,270 vulnerable contracts worth a total of 3,088,102 ETH, merely 49 contracts containing Ether may have been exploited for an amount of 9,066 ETH, which represents as little as 0.29% of the total amount at stake. Even if the rate is low, vulnerability always is a gateway for

TABLE IV
TAXONOMY OF PROGRAMMING LANGUAGE VULNERABILITIES [8], [40], [41], [42]

Languages Name	Vulnerabilities Name		Category
Solidity	Constant optimizer subtraction	EC recover malformed input	Operational
	ExpExponent Cleanup	Lopping through externally manipulated mappings or arrays	
	Send fails for zero ether	Costly loop	
	Dynamic allocation infinite loop	Clean bytes higher order bits	
	Delegate call return value	Event struct wrong data	Functional
	Floating point, unchecked division	Nested array function call decoder	
	Overflow/Underflow	Public lib functions do not return nested	
	Gasless send	Locked money	
	Libraries not callable from payable functions	Owner operations	
	Optimizer stale knowledge about SHA-3		
	Race condition	Transaction ordering dependence	Security
	Ether lost in transfer	Transfer forward all gas	
	Tx.origin	Unchecked external call	
	Identity precompile return ignored	Timestamp dependence	
	False randomness	Re-entrancy	
	External contract referencing	Uninitialized storage parameters	
	Array access clean higher order bits	High order byte clean storage	
	Optimizer clear state on code path join	Optimizer state knowledge not reset for jumpdest	
	Ancient compiler	Default visibility	Developmental
	Incorrect Interface	One of two constructors skipped	
	TypeCast/ Inference	Token API violation	
	Unpredictable state	Delegate call	Security / Funtional
	constructors with Care	Block Timestamp Manipulation	NA
	Denial Of Service(DOS)	Entropy Illusion	
	Short Address / Parameter Attack	Unchecked CALL Return Values	
	Floating Points and Numerical Precision		
C++	custom dispatcher	Dependence of transferring funds	NA
	persistent data on RAM	remote code execution	
	Buffer overflows	Dangling pointer references	
	Integer errors	Unchecked indirect calls	
	Denial of service attacks	NA	
Plutus	No Data	NA	NA
Ride	NA	NA	NA
Liquidity	NA	NA	NA
Sophia	NA	NA	NA
Kotlin	NA	NA	NA

TABLE V
VIRTUAL MACHINE BUGS AND VULNERABILITIES [44], [45], [34], [46], [47]

Virtual Machine	CVE-ID/Name	Published date	Vulnerability type
EVM	CVE-2018-18920	2018-11-03	No Data
	CVE-2018-19183	2018-11-11	DoS
	CVE-2018-19184	2018-11-11	DoS
	CVE-2018-19330	2018-11-17	No Data
	CVE-2019-7710	2019-02-10	No Data
	CVE-2017-14457	2018-01-19	DoS +Info
	Immutable bugs	No Data	No Data
	Ether lost in trasfer	No Data	No Data
	Stack size limit	No Data	No Data
WAVM	CVE-2018-16770	2018-07-26	DoS
	CVE-2018-16769	2018-07-26	DoS
	CVE-2018-16768	2018-07-26	DoS Overflow
	CVE-2018-16767	2018-07-26	DoS Overflow
	CVE-2018-16766	2018-07-26	DoS
	CVE-2018-16765	2018-07-26	DoS Overflow
	CVE-2018-16764	2018-07-26	DoS
Tron VM	No Data	No Data	No Data
LLVM	Stack Clash	2017	DoS
Functional Typed Warded Virtual Machine (FTWVM)	No Data	No Data	No Data

TABLE VI
COMPARATIVE TABLE ON FORMAL VERIFICATION [14]

Formal model name	Verification Method	Compatibility	Conclusions
F* Framework [53]	Translation and type checking	Not for all smart contracts, syntax limitation	Promising if it is expanded to work for more of the syntax. At the moment, only 46 out of 396 smart contracts were successfully verified.
Isabelle/HOL Framework [61]	Theorem proving	Not for all smart contracts, syntax limitation	Promising framework for use with a theorem prover, but does not support all Solidity syntax.
Model-Checking [54]	Model checking	Not for all smart contracts, syntax limitation	Promising approach that also models the blockchain environment itself. However, the modelling language can not translate all Solidity syntax.
Formal verification based on users and blockchain behaviours models [51]	Model checking	Not for all smart contracts, mostly relevant to contracts with human interaction or influence	Presents a new way to model smart contracts. This method is applied successfully to a specific contract, but it is uncertain if it is also applicable to other kinds of contracts.
Validation of DSC Through Game Theory and Formal Methods [52]	Model checking	Not for all smart contracts, mostly relevant to contracts with human interaction or influence	The addition of game theory enables the modeling of user behaviour in smart contracts, and adds a dimension to the formal verification. However, this approach is only focused on contracts that require human interaction.
Formal Verication of Deed Contract in Ethereum Name Service [60]	Symbolic execution	For all smart contracts	This approach proved the existence of security bugs, but did not formally verify their absence. Out of 19,366 smart contracts, 8,833 had the vulnerabilities that were tested for.

TABLE VII
OVERVIEW OF SMART CONTRACT ACCESS CONTROL MECHANISM

Mechanisms	Types	Algorithm-based	Model-based
Privacy based decentralized Public Key Infrastructure (PKI) [62]	authentication	PKI	User Based
Town Crier (TC): Authenticated Data Feed system [63]	Authentication / Authorization	Mathematical calculation	Data Based
Federated Identity Management without Third Party Authentication Services [64]	Authentication	PKI	User Based
Role-Based Access Control (RBAC) framework [65]	Authentication / Authorization	Mathematical calculation	User and Data Based
Enforcing Private Data Usage Control [1]	Authentication / Authorization	Mathematical calculation	User and Data Based
Multi-Authority Attribute-Based Access Control [66]	Authentication / Authorization	Mathematical / cryptographic calculation	User and Data Based

attacks. The aggravating factor of the problem of security is the immutability [74] of the application of blockchain and to this is added the phenomenon of the criminal smart contracts.

Many solutions have been proposed for the security of smart contracts, ranging from better development environments to better programming languages to formal verification and symbolic execution, and such tools are being developed. Advances in smart contract security will necessarily be layered, incremental, and necessarily dependent on defense-in-depth. There will be other bugs, and other lessons will be learned; there will not be a single magic technology that will solve everything [73]. To solve the security problem related to smart contracts it is important to find answers to these questions:

Q1: What are the solutions to correct vulnerabilities on an already deployed smart contracts?

Certainly the detection of vulnerabilities is of great importance for future smart contracts, the strong point of detection is correction. So far there is no deployed smart contract correction system . So we recommend future work on the vulnerability correction mechanisms of deployed contracts. To solve the problem of contracts already deployed, additional work could be carried out to revolutionize blockchain technology, that is to say, move to a new version (blockchain 3.0) which will allow us to go from a static state to a dynamic state of smart contract. This new state of smart contracts (i.e. dynamic) will make the task easier, it can either enable the sending of update of vulnerabilities correction of smart contracts or the evolution of vulnerability detection tools by integrating the correcting function after detecting vulnerabilities.

Q2: What technologies are likely to provide answers to the security problems of blockchain and transactions in general?

In addition, we have briefly discussed security levels on smart contract systems in general to classify them. Future project could consist in integrating into the blockchain, technologies such as artificial intelligence, deep learning or machine learning [75] and ontology [76].

Q3: Can advances in artificial intelligence be beneficial to blockchain and cryptocurrencies in terms of data and transaction security?

So far, cybersecurity systems using artificial intelligence have proven to be the most effective in protecting blockchain.

V. CONCLUSION

The prime objective of smart contracts is to strengthen the security of transactions on the blockchain but the fact

TABLE VIII
OVERVIEW OF ALL VULNERABILITIES DETECTING TOOLS INDICATING CODE LEVEL, TYPE AND ANALYSIS METHOD [55] [72]

	Level		Type		Analysis method					
Tools	Bytecode	Solidity code	Static analysis	Dynamic analysis	Code instrumentation	Symbolic execution	Constraint solving	Abstract interpretation	Horn logic	Model checking
contractLarva	✗	✔	✗	✔	✔	✗	✗	✗	✗	✗
E-EVM	✔	✗	✔	✗	✗	✗	✗	✗	✗	✗
Erays	✔	✗	✔	✗	✗	✗	✗	✗	✗	✗
EthIR	✔	✗	✔	✗	✗	✔	✔	✗	✗	✗
EtherTrust	✔	✗	✔	✗	✗	✗	✔	✔	✔	✗
FSolidM	form.spec		✔	✗	✗	✗	✗	✗	✗	✔
KEVM	✔	✗	✔	✗	✗	✗	✗	✗	✗	✗
MAIAN	✔	✗	✔	✔	✗	✔	✔	✗	✗	✗
Manticore	✔	✗	✔	✗	✗	✔	✔	✗	✗	✗
Mythril	✔	✗	✔	✗	✗	✔	✔	✗	✗	✗
Osiris	✔	✗	✔	✗	✗	✔	✔	✗	✗	✗
Oyente	✔	✗	✔	✗	✗	✔	✔	✗	✗	✗
Porosity	✔	✗	✔	✗	✗	✗	✗	✗	✗	✗
Rattle	✔	✗	✔	✗	✗	✗	✗	✗	✗	✗
Remix-IDE	✗	✔	✔	✗	✗	✗	✗	✗	✗	✗
Securify	✔	✗	✔	✗	✗	✗	✗	✔	✔	✗
SmartCheck	✗	✔	✔	✗	✗	✗	✗	✗	✗	✗
Solgraph	✗	✔	✔	✗	✗	✗	✗	✗	✗	✗
SolMet	✗	✔	✔	✗	✗	✗	✗	✗	✗	✗
Vandal	✔	✗	✔	✗	✗	✔	✗	✔	✔	✗
Ether	✔	✗	✔	✗	✗	✗	✗	✗	✗	✗
Gasper	✔	✗	✔	✗	✗	✔	✔	✗	✗	✗
ReGuard	✔	✔	✗	✔	✗	✗	✗	✗	✗	✗
SASC	✗	✔	✔	✗	✗	✔	✔	✗	✗	✗
sCompile	✗	✔	✔	✗	✗	✔	✔	✗	✗	✗
teEther	✔	✗	✔	✔	✗	✗	✔	✗	✗	✗
Zeus	✗	✔	✔	✗	✗	✗	✔	✗	✔	✗

is that smart contracts are often a source of problems. This survey presents an analysis of the security of smart contracts by studying the strengths and weaknesses. Our analysis is done based on academic literature, contribution to Internet blogs and discussion forums about Smart Contracts, and our experience on security. The paper highlights that the factor responsible for weak points in the security of smart contracts is the vulnerability factor that is the gateway to attacks. In addition to vulnerabilities, one of the common causes of smart contracts insecure is the difficulty of detecting inconsistencies between smart contract behavior and that of criminal smart contract. Although analysis and verification tools, and method based on Q-learning to invalidate criminal smart contracts [3] and the smart contract repairing framework [77] can help in this direction but are far from solving the problem. We expect editable smart contracts after deployment and this will make it easier for us to fix vulnerabilities.

Our future work will focus on setting up an intrusion detection system. The objective is to propose a consensus mechanism based on transaction behavior, the principle of which will be based either on the prototype of attacks linked to known vulnerabilities or on the normal behavior of a transaction. The idea is to propose a consensus algorithm that will prohibit nodes from responding to any transaction whose prototype is that of an attack or from responding to any transactions whose behavior is different from the normal behavior of a smart contract.

REFERENCES

[1] F. Glatz. (2014) What are smart contracts? https://heckerhut.medium.com/whats-a-smart-contract-in-search-of-a-consensus-c268c830a8ad.
[2] S. D. Levi and A. B. Lipton, *An Introduction to Smart Contracts and Their Potential and Inherent Limitations*, 2018, https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/.
[3] L. Zhang, Y. Wang, F. Li, Y. Hu, and M. H. Au, "A game-theoretic method based on q-learning to invalidate criminal smart contracts," *Information Sciences*, vol. 498, pp. 144–153, 2019.
[4] H. T. Le, N. T. T. Le, N. N. Phien, and N. Duong-Trung, "Introducing multi shippers mechanism for decentralized cash on delivery system," *money*, vol. 10, no. 6, 2019.
[5] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2018, pp. 1–4.
[6] D. Perez and B. Livshits, "Smart contract vulnerabilities: Does anyone care?" *arXiv preprint arXiv:1902.06710*, 2019.
[7] A. Dika, "Ethereum smart contracts: Security vulnerabilities and security tools," Master's thesis, NTNU, 2017.
[8] W. Dingman, A. Cohen, N. Ferrara, A. Lynch, P. Jasinski, P. E. Black, and L. Deng, "Defects and vulnerabilities in smart contracts, a classification using the nist bugs framework," *International Journal*

of Networked and Distributed Computing, vol. 7, no. 3, pp. 121–132, 2019.

[9] J. J. Xu, "Are blockchains immune to all malicious attacks?" *Financial Innovation*, vol. 2, no. 1, p. 25, 2016.

[10] R. Modi, *Solidity Programming Essentials: A beginner's guide to build smart contracts for Ethereum and blockchain*. Packt Publishing Ltd, 2018.

[11] Y. Hu, M. Liyanage, A. Mansoor, K. Thilakarathna, G. Jourjon, A. Seneviratne, and M. Ylianttila, "The use of smart contracts and challenges," *arXiv preprint arXiv:1810.04699*, 2018.

[12] M. Alharby and A. van Moorsel, "Blockchain-based smart contracts: A systematic mapping study," *arXiv preprint arXiv:1710.06372*, 2017.

[13] R. Rawat, R. Chougule, S. Singh, S. Dixit, and G. B.-P. A. Kadam, "Smart contracts using blockchain," *International Research Journal of Engineering and Technology (IRJET)*, 2019.

[14] Y. Murray and D. A. Anisi, "Survey of formal verification methods for smart contracts on blockchain," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2019, pp. 1–6.

[15] C. Dannen, *Introducing Ethereum and Solidity*. Springer, 2017.

[16] M. N. Temte, "Blockchain challenges traditional contract law: Just how smart are smart contracts," *Wyo. L. Rev.*, vol. 19, p. 87, 2019.

[17] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial internet of things," *Journal of Software Engineering and Applications*, vol. 9, no. 10, p. 533, 2016.

[18] T. Sameeh. (2018) An overview of the most reliable cryptocurrency smart contract platforms. https://www.cointelligence.com/content/smart-contract-platforms-guide/.

[19] Kryptographe. (2018) Which are the top 5 smart blockchain based smart contract platforms? https://www.kryptographe.com/top-5-smart-blockchain-based-smart-contract-platforms/.

[20] R. Jackson. (2019) The top 5 smart contract development platforms. https://hackernoon.com/top-5-smart-contract-platforms-to-check-out-in-2019-1igc3w1m.

[21] A. Davies. (2019) 5 best smart contract platforms for 2019. https://www.devteam.space/blog/5-best-smart-contract-platforms-for-2019/.

[22] N. Myers. (2019) The essential list of smart contract platform resources. https://www.freestartupkits.com/articles/technology/coding/the-essential-list-of-smart//-contract-platforms/.

[23] R. M. Parizi, A. Dehghantanha *et al.*, "Smart contract programming languages on blockchains: An empirical evaluation of usability and security," in *International Conference on Blockchain*. Springer, 2018, pp. 75–91.

[24] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, p. 37, 2014.

[25] Techopedia. (2019) Runtime environment (rte). https://www.techopedia.com/definition/5466/runtime-environment-rte.

[26] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, "Security analysis methods on ethereum smart contract vulnerabilities: A survey," *arXiv preprint arXiv:1908.08605*, 2019.

[27] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50 759–50 779, 2019.

[28] K. Kovalenko. (2019) Investing in smart contract platforms. https://www.blog.nomics.com/essays/investing-in-smart-contract-platforms/ platform-usage.

[29] Sanbase. (2019) All assets. https://https://www.app.santiment.net/assets/all/.

[30] T. Sameeh. (2019) Dapps statistics. https://www.stateofthedapps.com/stats/platform/ethereum/new/.

[31] M. brings transparency. (2019) Dapps statistics. https://messari.io/screener.

[32] M. Academic. (2020 (accessed 2020)) Vulnerability (computing). https://academic.microsoft.com/.

[33] M. Giancaspro, "Is a 'smart contract'really a smart idea? insights from a legal perspective," *Computer law & security review*, vol. 33, no. 6, pp. 825–835, 2017.

[34] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *International Conference on Principles of Security and Trust*. Springer, 2017, pp. 164–186.

[35] K. Chatterjee, A. K. Goharshady, and Y. Velner, "Quantitative analysis of smart contracts," in *European Symposium on Programming*. Springer, Cham, 2018, pp. 739–767.

[36] V. Saini. (2018) Contractpedia: An encyclopedia of 40+ smart contract platforms. https://hackernoon.com/contractpedia-an-encyclopedia-of-40-smart-contract-platforms-4867f66da1e5.

[37] C. Details. (2019) The ultimate security vulnerability datasource. https://www.cvedetails.com/vulnerability-list.php/.

[38] M. Gogan. (2018) Smart contract security: What are the weak spots of ethereum, eos, and neo networks? https://www.technative.io/smart-contract-security-what-are-the-weak-spots-of-ethereum-eos-and-neo-networks/.

[39] K. Jing. (2019) Eos smart contract development security best practices. https://github.com/slowmist/eos-smart-contract-security-best-practices/blob/master/.

[40] NIST. (2019) The bugs framework (bf). https://samate.nist.gov/BF/Classes/KMN.html.

[41] Github. (2018) Comprehensive list of known attack vectors and common anti-patterns. https://github.com/sigp/solidity-security-blog/precision-vuln.

[42] F. Junis, F. M. W. Prasetya, F. I. Lubay, and A. K. Sari, "A revisit on blockchain-based smart contract technology," *arXiv preprint arXiv:1907.09199*, 2019.

[43] Y. Fu, M. Ren, F. Ma, Y. Jiang, H. Shi, and J. Sun, "Evmfuzz: Differential fuzz testing of ethereum virtual machine," *arXiv preprint arXiv:1903.08483*, 2019.

[44] Y. Fu, M. Ren, F. Ma, H. Shi, X. Yang, Y. Jiang, H. Li, and X. Shi, "Evmfuzzer: detect evm vulnerabilities via fuzz testing," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2019, pp. 1110–1114.

[45] C. Details. (2018) Webassembly virtual machine project : Security vulnerabilities. https://www.cvedetails.com/vulnerability-list/.

[46] M. Larabel. (2019) Llvm stack clash compiler protection is under review. https://www.phoronix.com/.

[47] S. blog. (2017 (accessed december 14, 2019)) Decentralized application security project. https://steemit.com/blockchain/@aetrnty/aeternity-s-smart-contracts.

[48] D. Schatz, R. Bashroush, and J. Wall, "Towards a more representative definition of cyber security," *Journal of Digital Forensics, Security and Law*, vol. 12, no. 2, pp. 53–74, 2017.

[49] G. O. Karame and E. Androulaki, *Bitcoin and blockchain security*. Artech House, 2016.

[50] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016, pp. 254–269.

[51] T. Abdellatif and K.-L. Brousmiche, "Formal verification of smart contracts based on users and blockchain behaviors models," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2018, pp. 1–5.

[52] G. Bigi, A. Bracciali, G. Meacci, and E. Tuosto, "Validation of decentralised smart contracts through game theory and formal methods," in *Programming Languages with Applications to Biology and Security*. Springer, 2015, pp. 142–161.

[53] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Zanella-Béguelin, "Short paper: Formal verification of smart contracts," in *Proceedings of the 11th ACM Workshop on Programming Languages and Analysis for Security (PLAS), in conjunction with ACM CCS*, 2016, pp. 91–96.

[54] Z. Nehai, P.-Y. Piriou, and F. Daumas, "Model-checking of smart contracts," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 980–987.

[55] M. Di Angelo and G. Salzer, "A survey of tools for analyzing ethereum smart contracts," in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*. IEEE, 2019.

[56] J.-L. Lanet, "Détection de vulnérabilités appliquéea la vérification de code intermédiaire de java card," Ph.D. dissertation, UNIVERSITÉ DE LIMOGES, 2016.

[57] NASA. (2020) What is formal methods? https://shemesh.larc.nasa.gov/fm/fm-what.html.

[58] C. M. Holloway, "Why engineers should consider formal methods," in *16th DASC. AIAA/IEEE digital avionics systems conference. Reflections to the future. Proceedings*, vol. 1. IEEE, 1997, pp. 1–3.

[59] B. CURRAN, *How Formal Verification Can Reduce Bugs & Vulnerabilities in Smart Contracts*, 2018, https://blockonomi.com/formal-verification-smart-contracts/.

[60] Y. Hirai, "Formal verification of deed contract in ethereum name service," *November-2016.[Online]. Available: https://yoichihirai. com/deed. pdf*, 2016.
[61] S. Amani, M. Bégel, M. Bortin, and M. Staples, "Towards verifying ethereum smart contract bytecode in isabelle/hol," in *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*. ACM, 2018, pp. 66–77.
[62] P. Sivakumar and K. Singh, "Privacy based decentralized public key infrastructure (pki) implementation using smart contract in blockchain," *technical report*, 2018.
[63] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*. ACM, 2016, pp. 270–282.
[64] P. Mell, J. Dray, and J. Shook, "Smart contract federated identity management without third party authentication services," *arXiv preprint arXiv:1906.11057*, 2019.
[65] J. P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12 240–12 251, 2018.
[66] H. Guo, E. Meamari, and C.-C. Shen, "Multi-authority attribute-based access control with smart contract," in *Proceedings of the 2019 International Conference on Blockchain Technology*. ACM, 2019, pp. 6–11.
[67] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh, "Empirical vulnerability analysis of automated smart contracts security testing on blockchains," in *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 2018, pp. 103–113.
[68] H. Wang, Y. Li, S.-W. Lin, L. Ma, and Y. Liu, "Vultron: catching vulnerable smart contracts once and for all," in *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results*. IEEE Press, 2019, pp. 1–4.
[69] B. Jiang, Y. Liu, and W. Chan, "Contractfuzzer: Fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2018, pp. 259–269.
[70] USLegal, *Access Control Mechanism National Security Law and Legal Definition*, 2019, https://definitions.uslegal.com/a/access-control-mechanism-national-security/.
[71] M. Thakur *et al.*, "Authentication, authorization and accounting with ethereum blockchain," Master's thesis, Helsingfors universitet, 2017.
[72] A. Dika and M. Nowostawski, "Security vulnerabilities in ethereum smart contracts," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 955–962.
[73] V. Buterin. (2016) Thinking about smart contract security. https://blog.ethereum.org/2016/06/19/thinking-smart-contract-security/.
[74] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, 2019.
[75] F. Scicchitano, A. Liguori, M. Guarascio, E. Ritacco, and G. Manco, "A deep learning approach for detecting security attacks on blockchain." in *ITASEC*, 2020, pp. 212–222.
[76] H. M. Kim, M. Laskowski, and N. Nan, "A first step in the co-evolution of blockchain and ontologies: Towards engineering an ontology of governance at the blockchain protocol level," *arXiv preprint arXiv:1801.02027*, 2018.
[77] L. Y. XIAO, A.-B. OMAR, L. DAVID, and R. ABHIK, "Smart contract repair," *arXiv preprint arXiv:1912.05823v1*, 2019.