**A Survey on Static Analysis of Smart Contracts in Blockchains**

Regonda Brahmashwini

Ca ‘Foscari University of Venice

**ABSTRACT**

Blockchains are incredibly popular these days and constantly evolving. Smart contracts are simple programs that are stored in the blockchain which implements trusted transactions without third parties and are used to run when certain conditions are met. The rapid development of this blockchain with smart contracts has unmasked many security issues and exposed to some attacks because of the vulnerabilities in the smart contracts which have led to awful losses. To deal with this problem, surveyed static analysis of smart contracts in blockchains from recent papers. Static analysis is one of the impressive ways to detect vulnerabilities in smart contracts. Although it is significant for studying security verification of the blockchain smart contract it is obvious it is still really fresh. For this survey selected 5 papers which static analysis of Ethereum smart contracts and provide much rich information about the Ethereum smart contracts. This survey gives different frameworks, tools, languages, and challenges to detect vulnerabilities. Firstly surveyed a static analyzer called Slither and secondly on different tools, techniques, and challenges for static analysis of Smart contracts in blockchain. In total selected 20 papers for the related topic and also for the index terms that deal with the survey topic.

**INDEX TERMS** Static analysis, smart contracts, blockchains, survey, Ethereum

**I. INTRODUCTION**

A blockchain technique was originally developed by a group of researchers and was originally intended to timestamp digital documents so that it is not possible to backdate them or to tamper with them. Blockchain technology was created by fusing already existing technologies like cryptography, PoW, and decentralized networks. Blockchain [4] is a distributed ledger that is completely open to anyone. They have an interesting property that once some data has been recorded inside a blockchain it becomes very difficult to change it. Computers these days are fast and can calculate hundreds of thousands of hashes per second. So, to reduce this, blockchains have something called PoW. This mechanism makes it very hard to alter with the blocks. Blockchains are also constantly evolving

.

The smart contract proposed by Nick Szabo [3] is a digital agreement. The creation of blockchain technology has peaked a lot of people’s interest. Soon they realized that the technology could be used for other things like storing medical records, creating digital notaries. As for the problem of vulnerabilities of smart contracts, there is much loss to security issues. Slither [2] is a static analysis framework that is designed to provide rich information about the Ethereum smart contracts.

Ethereum [1] is the second-largest cryptocurrency. It is the most common deployment platform for smart contracts. The security of these Ethereum smart contracts has been under scrutiny [5]. Most of the known attacks in Ethereum smart contracts involve a hacker exploiting the smart contract vulnerability by social engineering attacks. In the year 2017, a security vulnerability appeared in the multi-signature wallet parity of Ethereum which led to the theft of $30 million, and in 2018 a painful BEC attack has caused theft of about $900 million. This made me think about ensuring the security and reliability of smart contracts.

Regarding the security and reliability of smart contracts is to include two aspects must be made:

1. As smart contracts are a static program that has not been put in use and the correctness of this program must be pre-conditioned to ensure security and reliability.
2. Secondly, the security problems that come during the implementation of the smart contracts must be considered.

So, surveyed the research results related to the security verification of Ethereum smart contracts from recent years.

This paper is organized as follows. Part II presents the preliminary about blockchain technology, smart contracts, static analysis, and Ethereum, part III overview of related papers, part IV about a static analyzer Slither and also discussed the advantages and disadvantages, part V about the challenges, techniques, platforms, and languages to address vulnerabilities in blockchain smart contracts and also discussed advantages and disadvantages. Then conclude the survey in the last part.

**II Preliminary**

**1. BLOCKCHAIN**

Blockchain technology [6] is an innovative way to implement decentralization, that is it is a solution for the problem of centralization. I. A decentralized way of maintaining a ledger that is practically impossible to forge. Each block (Figure 1) is linked to the data of the previous block, we have a chain of blocks, so it is called a blockchain.

Previous block

Next block

Version

Header

Body

Transaction counter

Hash1

Hash2

Hash N

Tran 1

Tran2

Tran N

Root Hash

**FIGURE 1 Structure of block in the blockchain**

The blockchain system allows the creation, verification, and updating of records by everybody with the help of four elements

1. **P2Peer Network**: A network of computers that are equally privileged nodes. It’s open to anyone and everyone. This is basically what we already have today on the internet. We need this network so that we will be able to communicate and share with others remotely.
2. **Cryptography Algorithms**: It is the art of secure communication in a hostile environment. It allows me to verify messages and prove the authenticity of oneself messages, even when malicious players are around. We need Cryptography because of the Peer-to-Peer Network as anyone can participate in the network including bad actors [5]. It’s great that anyone can communicate but also to make sure that oneself communication comes through unaltered.
3. **Consensus Algorithm**: We need to agree on the rules on how we add a new block to the records. There are many types of consensus rules, in BTC, the consensus algorithm is known as PoW. Ethereum shifted from PoW to proof of taking and getting this, PoS is a big technical challenge and is not a straightforward way as Pow across the network.
4. **Punishments and Rewards to the users**: Communication in the Peer-to-Peer network happens securely and follows a set of rules for reaching consensus. When all these four elements work together, give rewards to people that help maintain the records and add new blocks. The reward is a token or coin that is each time a consensus has been reached and a new block is added to the chain. On the other hand, bad actors who try to trick or manipulate the system will end up losing the money they spend on computational power, or their coins can be taken away from them.

Even though the percentage of fruitful attacks is less BTG, ETC, and BCG suffered from 51 percent of the power attack which led to huge losses. So, more attention is required on the security of blockchains.

**2. SMART CONTRACTS**

Smart contracts which are also called distributed apps are very popular nowadays. The term “smart contract” was first used by Nick Szabo in 1997, long before BTC was created [3]. These smart contracts are just contracts in the real world that are completely digital. In fact, a smart contract is a tiny computer program that is stored inside a blockchain. By which it authorizes the contract terms in the agreement automatically without the third party, with this it cuts down the admins and saves a lot of services.

People trust smart contracts because they are stored on a blockchain and inherit interesting properties like they are immutable and distributed [7]. And being distributed means that the output of the contract is validated by everyone on the network.

Smart contracts can be programmed in Solidity language. This language is created for Ethereum and uses a syntax that resembles JavaScript. It’s worth noting that BTC also supports smart contracts although it’s a lot more limited when compared to Ethereum. But the outputs of the experiment suggest that in terms of usability and Solidity one should use usable language for a new developer to write the code but when it comes to security these new developers write code that makes it vulnerable, and this may be used by some bad attacker to cause financial damages. In 2016 Decentralized Autonomous organization smart contract was tampered with to steal nearly 50 million $ and this was due to a reentrancy problem, a recursive call was created to smart contracts function to draw money for Ethereum when the call was only called by the contract specification.

Even though smart contracts seem to be simple to design and understand, it is different from real situations as we move to more scalable smart contracts in the blockchain. Because using this vulnerable smart contract which leads to attacks has gradually increased. In the year 2018 Nikolic [8] used the Maian analysis tool to perform a security analysis of almost one million smart contracts in 10 seconds for one contract, this analysis helped him to know about the flags that around 34 thousand contracts are in a vulnerable state. So, the problem of security verification for smart contracts is needed.

**3. STATIC ANALYSIS**

There are a variety of tools and frameworks to find the vulnerabilities in the smart contracts that were developed which are based on static analysis. These tools are based on the program testing methods like fuzzing, symbolic execution, and static analysis. This is the most effective way to find the issues in smart contracts. It basically analyzes the disassembled version of the executed contract. And then that is transformed to the contract code into an internal representation to detect the general security vulnerabilities.

Properties that should be in static analysis for Ethereum smart contracts are:

1. **Robustness of Static analyzer**: It should analyze real-life code without crashing.
2. **Performance of the static analyzer**: Analyzation should be very fast and even for scalable contracts so it will become easy to add easily into integrated development environments
3. **Exactness**: It should be able to allow for the developers to detect the potential problems while also should be able to maintain a low false positive. Like if the rate of false positives is more then it requires manual checking for the entire contract.
4. **Accurate level for the abstraction**: If the tool is too abstract it can be difficult to introduce correct semantics for capturing the general use patterns. In contrast, if the tool is too narrow in the detection of specific issues, then it will be hard to add new analyses.

At present Ethereum and BTC is the famous cryptocurrency mainly Ethereum which has a hard set of rules with scripts called smart contracts. As Ethereum smart contracts have millions of dollars, their execution correctness is very important against hackers who steal important assets. So with the help of SASC [1, 16] a static analysis tool that generates topology diagrams to find logical risks to the developers to understand the template clearly.

**4. Ethereum**

It is the second-largest cryptocurrency in the world after BTC. Forbes says it is the first generic blockchain that allows users to create and deploy trustless and decentralized applications [11]. In the past few years, different tools and frameworks were used to analyze and detect vulnerabilities for Ethereum smart contracts in blockchain with the help of static analysis [2].

To create a decentralized program that no single person controls, even the person who wrote it by using Ethereum programming called Solidity. The Ethereum platform has a number of independent computers running it. Once a program is deployed to the Ethereum network those computers will make sure it executes as written. By using Solidity coding language is used to write smart contracts that are the logic that runs decentralized applications.

An open-source framework Slither is used to provide Solidity information about Ethereum smart contracts. Nowadays the growth of smart contracts blockchain technology has put the main interest in cryptocurrency more on Ethereum because of its virtual machine [13] which supports smart contacts in a distributed way and that has control flow of ETH digital currency. There are many programming bugs in the Ethereum smart contracts [14].

**III OVERVIEW**

To have an overview of the static analysis of smart contracts in blockchain,

made a survey. Primarily, made a survey about static analysis of smart contracts in blockchain from Spring Link, ScienceDirect, ResearchGate, and IEEE Xplore by using the Index terms of static analysis along with blockchain and smart contracts.

Major contributions of this survey 5 related papers are selected as the most related studies. Secondly, analyzed a few papers related to the index terms mentioned in the survey. I proposed a taxonomy related to my survey topic as below

1) The aspect of a static analyzer called Sliter for smart contracts in blockchain covered by one paper [2] and that has 5 main use cases that include automated spotting of vulnerabilities and code optimization opportunities, improving user understanding of the smart contracts, assistance with the code review and implementation. And comparing Slither with another tool called Surya.

2) The aspect of the approaches and challenges to address vulnerabilities in blockchain smart contracts using automated tools and also frameworks along with the types of the languages used in four papers [1, 7, 12, 13].

The contribution of these 5 major papers is presented in Table 2

**IV a) Slither**

**A Static Analyzer for Smart Contracts in Blockchain**

Sliter is a static analysis framework that is designed to provide powerful information about the smart contract code for Ethereum. It uses the program's analysis techniques such as data flow and tracing. SlithIR is an intermediate representation of the Solidity of smart contracts. It uses a reduced instruction set along with the SSA for analyzing the ease of implementing the soundness to the bytecode (intermediary code in between the machine code and source code).

The contracts in Ethereum smart contract are written using the language Solidity, this contract is then compiled using EVM. Slither analysis is one of the most successful ways to detect issues in contracts. SlithIR is designed for practical analysis of Solidity code.

**Solidity code example:**

using SafeMath for uint;

mapping (address => uint) balances;

function transfer (address to, uint value)

public

{

balances[msg.sender] = balances[ msg.sender] .min(value);

balances[to] =balances[to]. add(value);

}

**Below is the code written in the Solidity language for SlitherIR for above code:**

Function transfer (address, uint256)

Solidity : balances[msg.sender] = balances [msg.sender.]. sub(value)

SlitherIR:

REF\_00(uint256) -> balances [msg.sender]

REF\_11(uint256) -> balances [msg.sender]

TMP\_1(uint256) =LIB\_CALL SafeMath.sub(REF\_11, value)

REF\_00 := TMP\_1(UINT256)

Solidity: balances[to] =balances[to].add(value)

SlitherIR:

REF\_33(uint256) -> balances [to]

REF\_44(uint256) -> balances [to]

TMP\_3(uint256) =LIB\_CALL, dest : SafeMath.add(REF\_44, value)

REF\_33 := TMP\_3(uint256)

Slither analyzer has five potential uses

1. Understanding the code for the users by summarizing and displaying. Slither analyzer provides printers that allow users to understand more quickly what a contract does and also how it is constructed. Its printer conveys a variety of representations based on graph structure like CFG, call graph, and inheritance graph for each of the contracts.
2. Code optimization detection automatically which complier losses [9]. The analyzer detects for the variables are declared like constants and these constants are optimized by the compiler which consumes less gas and that will not take space in the storage.

variable => constants => optimized by compiler

Functions must be declared as externals and these external functions allow the compiler to optimization of the code

functions=> external function=>this allow the compiler to optimize the code

1. Serve of reviewing the code for the user by application programming interface for the user to get interact with the analyzer. With the help of this API, users can design 3rd party tools and also scripts for using Slither.
2. Detection of bugs automatically by providing many types of smart contract vulnerabilities to detect the bugs without the user involvement by providing more than 20 bug detectors such as shadowing, uninitialized variables, reentrancy, and other security problems like suicidal contracts and sending arbitrary ETH. For example, say a user can check a particular variable that has never been spoiled by the parameter of the given function.

With the help of a third party.

1. Slither supports continuous integration and development toolboxes like Truffle and one more is Remix. It is implemented around 16000 lines of Python 3 and has fewer dependencies which only relies on the latest version of the strong compiler under the analysis.

Comparing slither with Survya:

As slither provides different tools for understanding code in the form of visual outputs like CFG and call graph, Survya is another tool with alike features and resolves the contract using Abstract Syntax Tree and also provides set out outputs. Slither provides information that is provided by Surya and also integrates more advanced information for an in-depth understanding of code.

For a better understanding of Surya and Slither by below **TABLE 1**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Category | Surya | Slither |
| 1 | Regarding Report | It provides mdreport | human report |
| 2 | Abstract Syntax Tree | It parses | NO |
| 3 | Regarding Function call trace | It provides fttrace | NO |
| 4 | Summary of functions | NO | YES |
| 5 | Regarding call graphs | ONLY Graph | YES |
| 6 | Access Summary and Authorization | NO | VARS, AUTH |
| 7 | Regarding contract summary | YES | YES |
| 8 | Regarding inheritance graph | ONLY Inheritance | YES |
| 9 | Regarding inheritance dependencies | ONLY Dependencies | ONLY Inheritance |
| 10 | Output | Text Representation of Abstract syntax tree | Variables Read and write |

**DISCUSSION ABOUT SLITHER**

Here summarized the advantages and weaknesses of the SLITHER tool which can be used as guidance for future studies.

Advantages

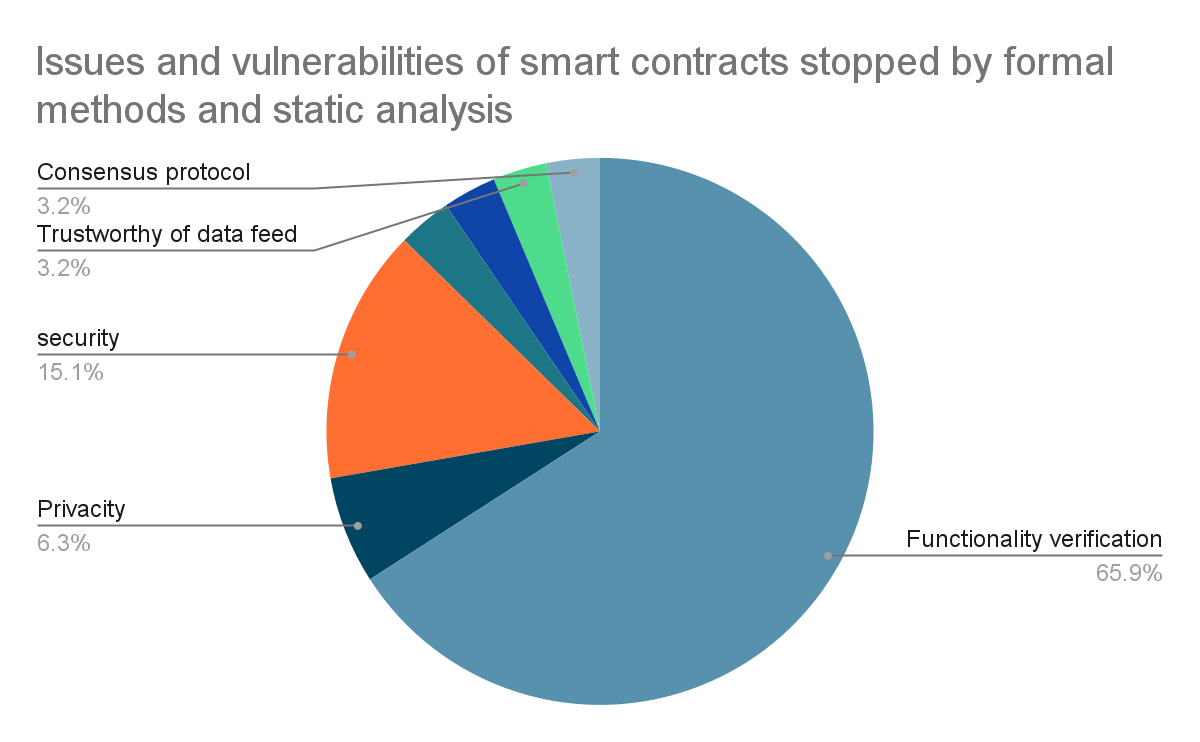
* Using SSA makes data dependency analysis straightforward.
* Symbolic execution built on the top of SlithIR representation allows easy access to detect bugs.
* SlithIR representation inside the Solidity compiler is a good step for a safer language.
* With the help of call graphs to understand the programs in a better way.

Disadvantages

* SlithIR has some limitations so still has room for development.
* It also lacks formal semantics which will allow more accurate analyses.
* As its representation is too high level it reflects the level of information.
* Solidity compiler fails to provide API on top of third-party analyzers.
* It fails to find true positives as the security flags are very small numbers of contracts when compared to other tools.

**V Approaches and challenges to address the vulnerabilities in the blockchain smart contracts**

A static analyzer is a framework that generates data flow for a piece of code and after that formal methods can be applied to check the system units. Symbolic executing a software testing technique will aid tests based on the data production and evidence regarding the quality of a program is important [1]. This kind of test result will not lead to attacks on smart contracts by an attacker. Additionally providing security assurance and the correctness of the smart contracts deserves more observation. As there is no limit of programming languages that can be used for smart contracts there is a wide scope for the attackers to attack the blockchain smart contracts. So this part will describe more about how to write reliable code for smart contracts.



**FIGURE 2**

A symbolic way of execution was used by Zhou in advanced static analysis frameworks for smart contracts in logical analysis for getting topological relationship graphs. And this symbolic executing static analyzer framework helps to find powerful risk for topological graphs [1] with relationships. Along with formal methods and symbolic execution, model checking is another way to check for the correctness and static analysis of blockchain smart contracts. To face real-world issues and check them use control flow along with the above-mentioned methods.

A framework named Osiris is used to find the integer bugs in the Ethereum smart contracts with the help of symbolic execution and taint analysis. It detects a greater range of bugs when compared to all existing frameworks with better detection. Vulnerabilities related to integer bugs are particularly important and difficult to ignore due to the few characteristics of EVM and the Solidity of programming languages.

Smart check is a static analysis tool that translates the powerful source code into XML intermediate representation and verifies against XPath patterns. This tool works in the real world big dataset by manual audit for three contracts and results in the present state knowledge on the solidity vulnerabilities and shows important improvement. But one disadvantage is that it can only detect some bugs.

**Discussion on different frameworks, languages, and challenges to detect vulnerabilities in a smart contract in the blockchain**

Blockchain Smart contract challenges as they have a new wave of invention in the business process. There are challenges that need to be tackled [7]. Firstly need to identify the life cycle of smart contracts. Address technical challenges and identify typical smart contract platform Challenges in creation, deployment, execution, and completion.

Advantages

* Frameworks will provide a security analysis of Ethereum smart contracts
* Developing smart contracts with safer language can help to eliminate a few vulnerabilities at the language level and the remaining few bugs can be detected using static analysis.
* Static analysis helps to fix the bugs in the smart contracts.
* If you use traditional high-level languages such as C++ AND JAVA to develop smart contracts it helps to support verification because it is much more familiar to developers.

Disadvantages

* Due to technical and legal logic, activities for smart contracts result in the features of ordinary contracts.
* Even after using stronger languages, powerful frameworks most of the time it detects only known bugs.
* As there is no limit for the language to be used to develop smart contracts it opens the doors for the attackers.

There are a few open challenges for the analysis of smart contracts in the blockchain those are

Firstly, the uncertain nature of the contract behavior, language syntax is not stable and lacks a clear property definition [12]. As smart contracts are designed based on the language for example Solidity for Ethereum smart contract will help more for the static analysis purpose. And also type-based approaches can help in a more expressive way to analyze smart contracts. Another alternative way could be the use of Machine learning to analyze input and to give potential compiler bugs.

**Table 2 Category and contribution of the major papers presented in this survey**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Reference Number | Category | Year | Contribution |
| 1 | [2] | Slither a static analyser | 2019 | It is an open source, fast,accurate static analyzer for smart contract blockchain |
| 2 | [1] | Ways and challenges to tackle the vulnerabilities in the blockchain using formal methods and static analysers. | 2020 | Different Methods for checking correctness and static analysis in smart contracts. |
| 3 | [12] | Techniques to analyze smart contract in the block chain | 2020 | Using static analysis to find Vulnerability detection and program correctness. |
| 4 | [13] | Frameworks for the static analysis for Ethereum smart contracts | 2018 | Presented verification tools. |
| 5 | [7] | Smart Contract challenges and platforms | 2020 | Automated analysis for EVM |

Smart contracts suffer from different types of vulnerabilities that result in financial losses. So many static analysis tools and techniques have been developed to detect vulnerabilities. With the help of these static analysis tools that only detect vulnerabilities known are “exploitable vulnerabilities” so-called only known vulnerabilities [12].

**Q1) What are the different languages that help to improve the security in the smart contracts in blockchain and that in turn helps static analysis easily?**

Languages used to formalize smart contracts in blockchains are Lem, SMACE, dsLAC, simplicity, Eitherlite, SCILLA, B Language, PROMELA, Securify language, SPECE, Findel. In this list one of the interesting ones for Ethereum smart contracts is ETHERLITE, an extract of EVM and a more interesting language is Securify language. It is a domain-based language for security platforms. This language is used for showing different security patterns for smart contracts [14, 20].

**Q2) What are the different Static analyzers that can be used to analyze Smart contracts in the blockchain?**

And there is a list of Static analyzers for smart contracts in blockchain they are Raziel, Mythril, Securify, Hawk, Contractlarva, SmartCheck, Oyente, KEVM, Theorem proving, ZEUSGlobal system safety framework, SASC, EtherTrust, Hydra, FsolidM, Slither, Surya, Symbolic Execution and MANIN. Significant frameworks for Ethereum Mytrill, contractLava, FsolidM, Osiris and eThor [1, 2, 13, 15, 17, 18, 19].

Q3) **What are the different platforms for smart contracts in the blockchain that in turn helps for static analysis?**

There are different platforms for smart contracts such as Ethereum, Fabric, Corda, Stellar, Rootstock, and EOS. Each platform has a different Executing environment like EVM, Docker, JVM, VM, and web assembly. And use different languages on different platforms and use different consensus [7].

The paper [1] gives many tools that can be used for static analyses in smart contracts to check functionality. Languages like Simplicity will help for the upper bound for static analysis for the smart contracts. SASC uses a formal verification method for the static analysis and can find logic risks and gives topological charts with relationships.

**CONCLUSION**

Smart contracts are one of the significant features in the blockchain which also has been exposed to many problems. The major contribution of the survey includes two aspects. Firstly, made a survey about the security verification of the blockchain in smart contracts with the help of Slither analyzer by selecting one paper related to it. Secondly surveyed one more paper to address the vulnerabilities in the blockchain using different tools and languages.

Important points are as follows:

1) According to the two major papers in the survey, can see the correctness and security of blockchain smart contracts mainly for Ethereum are getting more attention. Due to development in technology, day-by-day needs comprehensive methods to ensure the security of blockchain smart contracts by using powerful static analyzers for reducing losses.

2) For the security of blockchain smart contracts checking vulnerability is widely used to achieve important results. Even there are many good analyzers that do that job but still have unidentified vulnerabilities.

3) For the correctness of blockchain smart contracts more work is focused on programming for improving security.

4) Static analyzers are the essential part for the smart contract developers which helps in fixing bugs faster and also should allocate more effort to fix complex vulnerabilities.

**Acronyms**

1. Pow: Proof-of-Work
2. PoS: Proof-of-Stake
3. EVM: Ethereum Virtual Machine
4. ETH: Ether

1. BTC: Bitcoin

1. SSA: Static single assessment

1. BEC: Business Email Compromise

1. BTG: Bitcoin Gold

1. ETC: Ethereum Classic

**REFERENCES**

[1] Singh, Amritraj, et al. "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities." *Computers & Security* 88 (2020): 101654.

[2] Feist, Josselin, Gustavo Grieco, and Alex Groce. "Slither: a static analysis framework for smart contracts." *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2019.

[3] Szabo, Nick. "The idea of smart contracts." *Nick Szabo’s papers and concise tutorials* 6.1 (1997): 199.

[4] Nofer, Michael, et al. "Blockchain." *Business & Information Systems Engineering* 59.3 (2017): 183-187.

[5] Ivanov, Nikolay, et al. "Targeting the weakest link: Social engineering attacks in ethereum smart contracts." *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 2021.

[6] Zheng, Zibin, et al. "An overview of blockchain technology: Architecture, consensus, and future trends." *2017 IEEE international congress on big data (BigData congress)*. IEEE, 2017.

[7] Zheng, Zibin, et al. "An overview on smart contracts: Challenges, advances and platforms." *Future Generation Computer Systems* 105 (2020): 475-491.

[8] Nikolić, Ivica, et al. "Finding the greedy, prodigal, and suicidal contracts at scale." *Proceedings of the 34th annual computer security applications conference*. 2018.

[9] Chen, Ting, et al. "Under-optimized smart contracts devour your money." *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2017

[10] Matsuo, Shin'Ichiro. "How formal analysis and verification add security to blockchain-based systems." *2017 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2017.

[11] Mohanty, Debajani. "Ethereum Architecture." *Ethereum for Architects and Developers*. Apress, Berkeley, CA, 2018. 37-54

[12] Kim, Shinhae, and Sukyoung Ryu. "Analysis of blockchain smart contracts: Techniques and insights." *2020 IEEE Secure Development (SecDev)*. IEEE, 2020

[13] Grishchenko, Ilya, Matteo Maffei, and Clara Schneidewind. "Foundations and tools for the static analysis of ethereum smart contracts." *International Conference on Computer Aided Verification*. Springer, Cham, 2018

[14] Harz, Dominik, and William Knottenbelt. "Towards safer smart contracts: A survey of languages and verification methods." *arXiv preprint arXiv:1809.09805* (2018).

[15] Grishchenko, Ilya, Matteo Maffei, and Clara Schneidewind. "Ethertrust: Sound static analysis of ethereum bytecode." *Technische Universität Wien, Tech. Rep* (2018).

[16] Zhou, Ence, et al. "Security assurance for smart contract." *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2018.

[17] Torres, Christof Ferreira, Julian Schütte, and Radu State. "Osiris: Hunting for integer bugs in ethereum smart contracts." *Proceedings of the 34th Annual Computer Security Applications Conference*. 2018.

[18] Schneidewind, Clara, et al. "ethor: Practical and provably sound static analysis of ethereum smart contracts." *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020.

[19] Tikhomirov, Sergei, et al. "Smartcheck: Static analysis of ethereum smart contracts." *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*. 2018.

[20] Sergey, Ilya, et al. "Safer smart contract programming with Scilla." *Proceedings of the ACM on Programming Languages* 3.OOPSLA (2019): 1-30.