

Ethereum Smart Contract Threats, Attacks and Solutions: A Survey

Mr.Sachin S. Lamkane¹ and Dr.Ajay U. Surwade²

¹Department of Computer Science, Mudhoji College, Phaltan
Email: sslamkane2015@gmail.com

²Associate Professor, School of Computer Sciences, KBCNMU, Jalgaon
Email: ajaysurwade@gmail.com

Abstract—The importance and popularity of Ethereum smart contract has been continuously growing despite the availability of alternatives such as polkadot, Hyperledger Fabric and Stellar. Smart contracts are program that can be run on decentralized blockchain network with its most important property immutability. Due to this property, smart contract once deployed on the blockchain network, it cannot be modified or updated even for security enhancement purpose also. Hence it is necessary to develop strong secure contract program before deploying it on blockchain network and avoid potential exploitation and delay. Smart contracts are vulnerable to different typed of threats and attacks. To develop secure smart contracts is one of the most important challenges. In this paper we have analyzed different types of vulnerabilities on ethereum smart contract. We also review some available tools that detect these vulnerabilities. The main objective of this paper is to identify the research gap after carrying out literature survey of smart contract security and propose a model to improve security of smart contract.

Index Terms— Blockchain, Ethereum, Smart Contract, Vulnerability.

I. INTRODUCTION

Ethereum is a decentralized blockchain platform establishes a peer-to-peer network that securely executes smart contracts. Ether is the transactional token that facilitates to perform operations on Ethereum network. Ether is the natal cryptocurrency of the ethereum platform. Ether is the second largest cryptocurrency amongst different cryptocurrencies. Now-a-days an ethereum smart contracts have boomed, it has become an integral part of the blockchain ecosystem. Actually the concept of smart contract is introduced by Nick Szabo in 1997 [1]. Smart contracts are programs that are hosted on decentralized blockchain network. The smart contract allows two or more parties to form an agreement and initiate transaction between these parties without an involvement of intermediary [2]. Ethereum is a blockchain platform which provides tools for developer to build decentralized application using Ethereum Virtual Machine (EVM). It enables involving parties to make an agreement in a transparent and secure way. However there exists some security vulnerabilities within these smart contracts and hence it causes huge financial loss. Vulnerability in smart contract cannot be fixed, due to the immutable property of blockchain. Hence smart contracts are targeted by different kinds of attacks. Most of the attacks on smart contract are exploited due to the poor smart contract code [3]. In fact, ethereum smart contract already faced several overwhelming attacks including Decentralized Autonomous Organization (DAO) attack, Parity

Multisig wallet attack. The DAO attack in 2016 [4] and the parity multisig wallet attack in 2017 [5], which together causing a loss of over \$400 million due to bugs in the smart contract code.

Various tools have been proposed so far for detecting the vulnerabilities before deploying the contracts, but still it lacks complete solution [6, 7, 8]. So it is needed to identify possible vulnerabilities and filtering it out before deploying it on blockchain. This paper is organized as motivation which is followed by literature review, followed by research gap identified which is followed by the proposed model, followed by conclusion and future work.

II. MOTIVATION

Ethereum smart contracts were introduced in 2015, since that there have been several attacks exploited on smart contract [9]. The motivations behind this study are some observations on ethereum smart contract attacks are as follows:

- The first attack was in 2016, when an attacker launched a reentrance attack against the Decentralized Autonomous Organization (DAO) smart contract, which resulted in a loss of more than 36,00,000 million ethers worth more than US\$60 million[10].
- Parity wallet, a digital storage service for cryptocurrency such as ethereum was hacked in 2017, with approximately \$34 million stolen. It is estimated that hackers have stolen a total of \$2 billion since 2017[11].
- In 2018, an adversary stole more than 165 ethers worth of almost US\$40,000 from spankchain due to reentrancy attack in the smart contract [10].
- In 2019, about 34,200 ethereum smart contracts worth \$4.4 million ether are vulnerable to hacking due to poor coding and containing bugs [12].
- In 2020, there were 47 successful attacks aimed at ETH DApps, which cost victims around \$436.36 million or \$ 9.28 million per hack [13].

The above statistics shows that, ethereum smart contract threats and attacks are increasing day by day. It has become important to propose tools to detect threats on ethereum smart contract.

III. LITERATURE REVIEW

1. Ethereum smart contract threats and attacks increasing day-by-day. Ethereum smart contracts are susceptible to different kinds of attack. Following are some major attacks on ethereum smart contract.
The DAO Attack – The contract DAO is a financial application running on top of Ethereum. In June 2016, the DAO hack occurred when the attacker managed to steal more than 3.6 million ethers [6]. The DAO attack caused by reentrancy vulnerability [17].
The reentrancy vulnerability exploitation in the DAO attack was accomplished in four steps.[17]
Step 1. The Attacker or Hacker initiates a transaction by calling withdraw function of victim.
Step 2. The victim transfers the money and calls the fallback function of the attacker.
Step 3. The fallback function recursively calls the withdraw function again i.e. reentrancy.
Step 4. Within an iteration, extra ether will be transferred multiple times to the attacker.
2. *Parity multisig wallet attack* – The parity multisig wallets are smart contract programs which are used to manage digital assets by the wallet users [38]. In ethereum, a multisig wallet requires multiple private keys to unlock a wallet. The parity multisig wallet attack was compromised twice in 2017. The first attack exploited the delegate call injection vulnerability and erroneous visibility vulnerability. The second attack exploited the unprotected suicide vulnerability and frozen ether vulnerability. This attack is the second largest attack on the ethereum network in terms of amount of stolen ether [39].
3. *King of Ether Throne Attack* – The king of the ether throne contract (KotET contract) exploit case of out-of-gas exception vulnerability. KotET contract is a game, where player complete to become the king by paying some ether to the current king, plus a small fee to the contract owner [40]. After the contract declares a new king of the ether throne, the new claim price for the throne goes up by 50%.
4. *Rubixi* – The Rubixi contract is a Ponzi scheme containing erroneous constructor name vulnerability. Rubixi is a contract which is a fraudulent high yield investment program where participant gain money from the investments made by new comers. Further, the contract owner can collect some fee paid to the contract upon investments [41].

5. *Governmental* – Governmental scheme attack exploit case of mishandled exceptions vulnerability. In this attack, a contract that implements a flawed Ponzi scheme [3]. This scheme requires a participant to send a certain amount of ether the scheme contract. If no one joins the scheme for 12 hours, the owner of the contract keeps his fee and transfers the remaining ether to the last participant.
6. *Reentrancy Honeypot* – Reentrancy honeypot caused due to Typecast Vulnerability. Honeypot contracts are deployed on the ethereum main network to capture ethereum hackers who try to exploit the contract. A small scale attack using the typecasts vulnerability was launched on a honeypot developed to capture hackers trying to exploit the reentrancy vulnerabilities in smart contract.
7. *Proof of Weak Hand Coin (PoWHC)* –Proof of Weak Hand coin (PoWHC) attack caused due to integer underflow/overflow vulnerability. An integer overflow / underflow occur when an arithmetic operation is performed that requires a fixed size variable to store data that falls outside the range of the variable [42]. The integer type unit 256 in solidity has a limited size up to 256 bits. If the value of integer variable reaches its maximum value as $2^{256} - 1$, then it will automatically reset to zero when an additional integer 1 is added to the variable. Hackers target these variables in smart contract to make vulnerable by increase or decrease the value of integers until they reach to the maximum or minimum value [43].
8. *HYIP* – The contract HYIP was another Ponzi scheme, which pays existing investors from funds contributed by new investor at the end of each day. The contract HYIP exploitation case of DoS by external call vulnerability.

The researchers have proposed different tools that identify different vulnerabilities in smart contract.

Table 1, shows review of the important contributions or solutions which have been proposed so far to stop different threats and attacks.

TABLE 1. IMPORTANT CONTRIBUTION

Authors	Dataset used	Techniques/ methodology	Contribution
Chen, W.et al. (2020) [14]	Total 857 addresses from HONEYBADGER Project	Machine learning model to detect honeypot contract based on N-gram features and LightGBM.	The authors have analyzed honeypot contracts on ethereum and build a LightGBM model to detect the honeypot contract.
W. Chen et al. (2018) [15]	-	The machine learning model is proposed	The authors have proposed a machine learning framework to detect smart Ponzi scheme, a classic scam under the veil of smart contract on ethereum.
Tann, W. J. W. et al. (2018) [16]	Experimental study on 620,000 smart contracts	The technique of machine learning long-short term memory (LSTM) is used to detect new attack trends.	The authors have proposed the approach of sequential learning of smart contract weaknesses using machine learning long-short term memory (LSTM) - that allows detecting new attack trends quickly. Accuracy reported 99.57% and F1 score 86.04%
Ivica Nikolic et al. (2018) [12]	Total 34200 contracts flagged	Static symbolic analyzer used to propose MAIAN tool	MAIAN tool for detecting Greedy, prodigal and Suicidal contracts. Results of these three contracts over given dataset are prodigal 97%, suicidal 99% and greedy 69% true positive.
Luu L et al. (2016) [17]	Total 19,366 existing contracts are tested. It takes two inputs Ethereum smart contract bytecode and ethereum global state.	Oyente is a symbolic execution tool aims to find potential security bugs.	The proposed oyente tool enhances the operational semantics of ethereum to make contracts less vulnerable. The proposed tool finds potential security bugs. It is a static analysis tool which detects the security vulnerability.

Kolluri A et al. (2019) [18]	-	The authors have developed an efficient analysis technique for ethereum smart contracts under multiple events. They investigate a family of bugs in smart contract called Event-Ordering (EO) bugs.	ETHRACER is an automatic analysis tool which runs directly on ethereum bytecode and requires no hints from users in order to detect event ordering bugs in blockchain smart contract.
Jiang B et al. (2018) [19]	Flagged fuzzing of 6991 smart contracts.	Presented novel fuzzer to test the vulnerabilities in ethereum smart contract	The author proposed ContractFuzzer tool which is novel fuzzer to test Ethereum smart contracts for security vulnerabilities. ContractFuzzer generates fuzzing inputs based on the ABI specifications of smart contracts, defines test oracles to detect security vulnerabilities, instruments the EVM to log smart contract run-time behaviors, and analyzes these logs to report security vulnerabilities.
Bragagnolo S et al. (2018) [20]	-	Decomposition technique and mirror based reflection used to analyse contract state	SmartInspect is a solidity smart contract inspector that aims at analyzing contract states using decompilation techniques driven by the contract structure definition. It also allows contract developers to better visualize and understand the contract stored state without needing to redeploy, nor develop any ad-hoc code
S. Kalra et al. (2018) [21]	-	Total six different types of vulnerabilities in smart contract were detected. Static analysis technique is used.	ZEUS verifies the correctness of smart contract and validates their fairness. The tool ZEUS detects six security vulnerabilities in smart contract.
T. Chen et al. (2017) [22]	Uses smart contract bytecode	Static analysis tool is used to propose GASPER tool	The author proposed GASPER tool to detect the smart contracts with inefficient gas consumption. Seven solidity code patterns were identified which are used by GASPER for detection purpose.
L. Brent et al. (2018) [23]	-	Analysis pipeline is used to convert EVM bytecode to the semantic logic relations.	The author proposed tool Vandal is a security analysis framework for identifying the vulnerabilities in ethereum smart contract. Vandal identifies security vulnerabilities such as unchecked send, re-entrancy, unsecured balance, destroyable contracts and use of origin problem.
E. Albert et al.(2018) [24]	Uses ethereum smart contract bytecode and ethereum global state	Rule based representation produced by OYENTE is used to propose Ethir tool.	The proposed tool analyses ethereum smart contract bytecode based on the rule based representations of the Control Flow Graph (CFG) produced by the OYENTE tool
P. Tsankov et al.(2018) [25]	The tool was evaluated on two datasets- the EVM dataset and solidity dataset.	Used two components- dependency graph and critical code structure.	The author proposed fully automated and scalable security analyzer for ethereum smart contract
Albert, E. et al.(2018) [26]	-	Uses five tools to generate closed form gas bands.	The platform developed to calculate the upper bond for the required amount of gas for ethereum smart contract to avoid the out of gas vulnerability.

Jiachi Chen et al.(2021) [27]	Applied on Open source Dataset. Tested on total 1,65,621 Smart contract	Proposed symbolic execution-based approach and tool to detect eight contract defects.	Author proposed DEFECTCHECKER is a symbolic execution based model for smart contract defect detection. These proposed models obtain high F-score on an open source dataset i.e. 88.8%.
Yu Feng et al.(2019) [28]	Evaluated on entire data set from etherscan with > 25K smart contracts	Proposed a summary based symbolic evaluation technique which reduces program size.	Proposed tool SMARTCOPY is an attack synthesizer which incorporates a novel combination of search space partitioning, parallel symbolic execution and early pruning based on the abstract semantic of candidate program. This proposed model obtains the FN and FP rates as 7% and 10% for timestamp vulnerability and 14% and 5% for Reentrancy vulnerability.
Sergei Tikhomirou et al.(2018) [29]	The proposed tool is evaluated on big dataset of real world contracts from etherscan.	Proposed an extensible static analysis tool to detect vulnerabilities in Smart contract.	Proposed SMARTCHECK model helps to remove the simple bugs quickly from smart contract. This tool showed better False Discovery Rate (FDR) and False Negative Rate (FNR) compared to its closest competitor Remix, Oyente and Securify. This tool showed overall FDR of 69% and Overall FNR 47%.
Torres, C. F et al. (2018) [30]	Evaluated on large experimental dataset containing more than 1.2 million smart contracts	The proposed model employs a strategy based upon taint analysis and symbolic execution.	Author introduced OSIRIS, a symbolic execution tool that automatically detects integer bugs in EVM bytecode. This tool reports less false positive rate as compared to other tool ZEUS. Overall evaluation on 1.2 million ethereum smart contracts indicates that about 4% of them might be vulnerable to at least one bug from arithmetic Bug, Truncation bug or Signedness bug.
Neville Grech et al. (2018) [31]	Validate 6.3 million contracts deployed on blockchain.	The proposed model is a static program analysis technique used to detect gas focused vulnerability in smart contract. This approach combines a control-flow-analysis based decompiler and declarative program structure queries.	Proposed model MadMax uses static program analysis directly on the EVM bytecode. This model identifies out-of gas vulnerabilities. The analysis on given dataset reports vulnerabilities for contracts holding a total value of over \$2.8B.
Everett Hildenbrandt et al. (2017) [32]	-	The proposed model is a semantic analysis tool based upon SE. Framework of this model integrates a semantic debugger and a program verifier.	Proposed KEVM model which is fully formally rigorous ethereum interpreter that does not rely on a built-in hard coded implementation of the EVM.
Anastasia Mavridou et al.(2018) [33]	-	Proposed model to help developers to create more secure smart contracts. A presented framework rooted in rigorous semantic for designing contracts as Finite State Machine(FSM)	Author presented a tool FSolidM for creating Finite State Machine easy interface auto generating ethereum contract. In this tool author introduce set of design patterns, as plugins that developers can easily add to enhance security and functionality of their contract.
Mavridou, A. et al. (2019) [34]	-	Proposed approach used formal verification tool	Author presented the VeriSolid framework, which introduces formal verification capabilities providing an approach for correct-by-design development of smart contract. Author design and implement developer friendly natural language for

			specifying safety and liveness properties. It prevents reentrancy as well as Denial of Service vulnerabilities.
Johannes krupp et al. (2018) [35]	-	Methodology used to find vulnerability is symbolic execution tool.	Author develops a tool TEETHER that provides end to end exploits generation from a contracts bytecode only. They provide a large scale vulnerability analysis of 38,757 unique contracts extracted from the ethereum blockchain and found exploits for 815(2.10%) of those completely automated.
Grishchenko, I. et al.(2018) [36]	Tested on real world contracts from the ethereum blockchain	Proposed static reachability analysis technique for EVM bytecode, that handles the complexity of EVM bytecode	Proposed tool EtherTrust, a static analyzer that analyzes large contracts in a few seconds. It is a tool provides security guarantee for EVM bytecode.
Liu, C. et al.(2018) [37]	-	The proposed model is based of fuzzing-based analyzer	Proposed tool ReGuard, a fuzzing-based analyzer that detects reentrancy bug in ethereum smart contract

IV. RESEARCH GAP IDENTIFIED

1) To identify importance of Ethereum in Block chain.

The importance of execution of Ethereum would be studied with the context of Block-chain. The role and importance of Ethereum would be investigated in the working of Blockchain.

2) To investigate the execution of Smart Contracts and susceptibility of Smart Contracts to different types of Attacks.

The Study of execution of Smart Contracts would be carried out. During this study main focus would be on the susceptibility of Smart Contracts to different types of attacks. The detail study of these attacks would be carried out.

3) To understand the working of Greedy contract Threat and it's impact on the working of Smart Contract.

One of the major attack Smart contract is suffering from is Greedy contract Threat. The study of Greedy contract Threat would be carried out along with the different dimensions including the way of attack, different methods used in executing this attack. Impact analysis of Greedy contract Threat on Smart Contract would be carried out.

4. To develop new technique or improve existing technique/es for detecting Greedy contract Threat.

After studying existing techniques or methods to deal with Greedy contract Threat existing technique/es would be improved or new technique/es would be proposed.

All these techniques are evaluated using evaluation metrics such as Accuracy, Precision, Recall and F-Score

V. IMPLEMENTATION DETAIL

Smart Contract testing is relatively complicated, as smart contracts often interact with each other, and any integration with a third-party system can lead to making the system vulnerable. Therefore, testing is often extended to other smart contracts that interact with the smart contract under test. The testing process includes static analysis, dynamic analysis and interaction analysis with other Smart Contracts, using advanced testing tools and manual reviewers. Methods used for vulnerability detection in smart contract include symbolic execution, formal verification, fuzzy testing, program analysis, taint analysis, ML-based methods RNN, LSTM, ABCNN. In order to implement the model for detecting vulnerability in ethereum smart contract advanced security scanning tools required to scan for vulnerabilities in smart contract. Following are steps to implement advanced security scanning tools for vulnerabilities detection in smart contract,

Step I To find the scope and content of work for the smart contract.

Step II Test the component in the smart contract, using advanced security scanning tools to detect common vulnerabilities.

Step III Simulate interactions with smart contract on the blockchain using a combination of automated testing tool to determine if any vulnerabilities exists.

Step IV Aggregate the result and check for accuracy, Precision, Recall and F-Score

After finding out the research gap from the literature survey an architecture has been proposed this is shown in Fig. 1, to classify greedy contract threat.

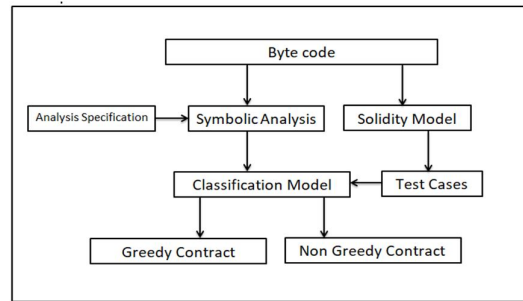


Fig.1 Architecture for greedy contract classification

VI. CONCLUSION

This proposed research addresses to the detection of vulnerabilities in smart contract and susceptibility of smart contract to different types of threats and attack. The primary objective of this paper is to find out research gap after carrying out literature survey of tools and techniques to find vulnerabilities in smart contract and identify greedy contract threat. The research gap identified then can be used to propose new technique for detecting vulnerabilities in ethereum smart contract which would be efficient and effective. Smart contract advanced security scanning tools would be developed to scan for vulnerabilities in smart contract.

REFERENCES

- [1] N. Szabo, The idea of smart contracts, Nick Szabo's Papers and Concise Tutorials (1997).URL http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vw.h.net/smart_contracts_2.html.
- [2] Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8, 24416-24427.
- [3] Atzei, N., Bartoletti, M., & Cimoli, T. (2017, April). A survey of attacks on ethereum smart contracts (sok). In *International conference on principles of security and trust* (pp. 164-186). Springer, Berlin, Heidelberg.
- [4] Mehar, M. I., Shier, C. L., Giambattista, A., Gong, E., Fletcher, G., Sanayhie, R., ... & Laskowski, M. (2019). Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack. *Journal of Cases on Information Technology (JCIT)*, 21(1), 19-32.
- [5] Petrov, S. (2017). Another parity wallet hack explained.
- [6] Siegel, D. (2018). Understanding the DAO attack (2016). URL <https://www.coindesk.com/understanding-dao-hack-journalists>, 41, 43.
- [7] Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., & Roscoe, B. (2018, May). Regard: finding reentrancy bugs in smart contracts. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)* (pp. 65-68). IEEE.
- [8] Breidenbach, L., Daian, P., Tramèr, F., & Juels, A. (2018). Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts. In *27th {USENIX} Security Symposium ({USENIX} Security 18)* (pp. 1335-1352).
- [9] Alkhalifah, A., Ng, A., Chowdhury, M. J. M., Kayes, A. S. M., and Watters, P. A. (2019). "An empirical analysis of blockchain cybersecurity incidents," in 2019 IEEE Asia-Pacific conference on computer science and data engineering (CSDE), Melbourne, Australia, December 9–11, 2019 (IEEE), 1–8. doi:10.1109/CSDE48274.2019.9162381
- [10] Alkhalifah, A., Ng, A., Kayes, A. S. M., Chowdhury, J., Alazab, M., and Watters, P. A. (2020). "A taxonomy of blockchain threats and vulnerabilities," in *Blockchain for cybersecurity and privacy: architectures, challenges and applications*. (Boca Raton, FL: CRC Press Taylor & Francis), Chap. 1, 1–27.
- [11] Accessed on date 10/07/2021 www.captchu.edu/blog/smart-contract-hacking-what-it-and-what-does-it-affect.
- [12] Nikolić, I., Kolluri, A., Sergev, I., Saxena, P., & Hobor, A. (2018, December). Finding the greedy, prodigal, and suicidal contracts at scale. In *Proceedings of the 34th Annual Computer Security Applications Conference* (pp. 653-663).
- [13] Accessed on date 10/07/2021 www.aithority.com
- [14] Chen, W., Guo, X., Chen, Z., Zheng, Z., Lu, Y., & Li, Y. (2020, August). Honeypot contract risk warning on ethereum smart contracts. In 2020 IEEE International Conference on Joint Cloud Computing (pp. 1-8). IEEE.
- [15] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 1409–1418

- [16] Tann, W. J. W., Han, X. J., Gupta, S. S., & Ong, Y. S. (2018). Towards safer smart contracts: A sequence learning approach to detecting security threats. arXiv preprint arXiv:1811.06632.
- [17] Luu L, Chu DH, Olickel H, Saxena P, Hobor A (2016) Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery, New York, pp 254–269.
- [18] . Kolluri A, Nikolic I, Sergey I, Hobor A, Saxena P (2019) Exploiting the laws of order in smart contracts. In: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis. Association for Computing Machinery, New York, pp 363–373.
- [19] Jiang B, Liu Y, Chan WK (2018) Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. Association for Computing Machinery, New York, pp 259–269
- [20] Bragagnolo S, Rocha H, Denker M, Ducasse S (2018) Smartinspect: Solidity smart contract inspector. In: 2018 International workshop on blockchain oriented software engineering (IWBOSE), pp 9–18
- [21] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, “Zeus: Analyzing safety of smart contracts,” in Proceedings of NDSS, 2018, Conference Proceedings.
- [22] T. Chen, X. Li, X. Luo, and X. Zhang, “Under-optimized smart contracts devour your money,” in Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on. IEEE, 2017, Conference Proceedings, pp. 442–446.
- [23] L. Brent, A. Jurisevic, M. Kong, E. Liu, F. Gauthier, V. Gramoli, R. Holz, and B. Scholz, “Vandal: A scalable security analysis framework for smart contracts,” arXiv preprint arXiv:1811.10403.
- [24] E. Albert, P. Gordillo, B. Livshits, A. Rubio, and I. Sergey, “Ethir: A framework for high-level analysis of ethereum bytecode,” in International Symposium on Automated Technology for Verification and Analysis. Springer, 2018, pp. 513–520.
- [25] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, “Securify: Practical security analysis of smart contracts,” in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018, pp. 67–82.
- [26] Albert, E., Gordillo, P., Rubio, A., & Sergey, I. (2018). GASTAP: A gas analyzer for smart contracts. CoRR, vol. abs/1811.10403.
- [27] Chen, J., Xia, X., Lo, D., Grundy, J., Luo, X., & Chen, T. (2021). Defectchecker: Automated smart contract defect detection by analyzing evm bytecode. IEEE Transactions on Software Engineering.
- [28] Feng, Y., Torlak, E., & Bodik, R. (2019). Precise attack synthesis for smart contracts. arXiv preprint arXiv:1902.06067.
- [29] Tikhomirov, S., Voskresenskaya, E., Ivanitskiy, I., Takhaviev, R., Marchenko, E., & Alexandrov, Y. (2018, May). Smartcheck: Static analysis of ethereum smart contracts. In Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (pp. 9-16).
- [30] Torres, C. F., Schütte, J., & State, R. (2018, December). Osiris: Hunting for integer bugs in ethereum smart contracts. In Proceedings of the 34th Annual Computer Security Applications Conference (pp. 664-676).
- [31] Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., & Smaragdakis, Y. (2018). Madmax: Surviving out-of-gas conditions in ethereum smart contracts. Proceedings of the ACM on Programming Languages, 2(OOPSLA), 1-27.
- [32] Hildenbrandt, E., Saxena, M., Zhu, X., Rodrigues, N., Daian, P., Guth, D., & Roşu, G. (2017). Kevm: A complete semantics of the ethereum virtual machine.
- [33] Mavridou, A., & Laszka, A. (2018, February). Designing secure ethereum smart contracts: A finite state machine based approach. In International Conference on Financial Cryptography and Data Security (pp. 523-540). Springer, Berlin, Heidelberg.
- [34] Mavridou, A., Laszka, A., Stachtari, E., & Dubey, A. (2019, February). VeriSolid: Correct-by-design smart contracts for Ethereum. In International Conference on Financial Cryptography and Data Security (pp. 446-465). Springer, Cham.
- [35] Krupp, J., & Rossow, C. (2018). teether: Gnawing at ethereum to automatically exploit smart contracts. In 27th {USENIX} Security Symposium ({USENIX} Security 18) (pp. 1317-1333).
- [36] Grishchenko, I., Maffei, M., & Schneidewind, C. (2018). Ethertrust: Sound static analysis of ethereum bytecode. Technische Universität Wien, Tech. Rep.
- [37] Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., & Roscoe, B. (2018, May). Reguard: finding reentrancy bugs in smart contracts. In 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion) (pp. 65-68). IEEE.
- [38] Breidenbach, L., Daian, P., Juels, A., & Sirer, E. G. (2017). An in-depth look at the parity multisig bug. URL: <http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/> (visited on 09/03/2018).
- [39] Destefanis, G., Marchesi, M., Ortu, M., Tonelli, R., Bracciali, A., & Hierons, R. (2018, March). Smart contracts vulnerabilities: a call for blockchain software engineering?. In 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE) (pp. 19-25). IEEE.
- [40] Thompson, S., Lamela Seijas, P., & Adams, D. (2016). Scripting smart contracts for distributed ledger technology.
- [41] Bitcointalk: Hi!My name is Rubixi, <https://bitcointalk.org/index.php?topic=1400536.60>
- [42] Accessed on 6-10-2021. Ethereum Home Page. <https://www.ethereum.org/>
- [43] Min, T., & Cai, W. (2019, June). A security case study for blockchain games. In 2019 IEEE Games, Entertainment, Media Conference (GEM) (pp. 1-8). IEEE.