Security issues in Preddy, A Blockchain Prediction Market

Eugene Koh
U1721183C
Renaissance Engineering Programme
Nanyang Technological University
Singapore
EKOH016@e.ntu.edu.sg

Milla Samuel
U1721667L
Renaissance Engineering Programme
Nanyang Technological University
Singapore
MSAMUEL001@e.ntu.edu.sg

Stephen Ng
U1721440G
Renaissance Engineering Programme
Nanyang Technological University
Singapore
STEP0036@e.ntu.edu.sg

***Abstract*—Blockchain Prediction Markets (BPMs) make use of blockchain technologies to create a low-cost, trustless and decentralized betting avenue. We present Preddy, a BPM that boasts custom distribution of initial liquidity on creation of a new market, as well as real time price, odds and earnings information. Preddy's unique features give rise to certain issues in security to be explored. Smart contract vulnerabilities, front-running and network attacks as well as the naivety of the dispute mechanism are the most pertinent security issues arising from the contract layer, network layer and application layer respectively. We recommend the use of an automated security analysis framework, submarine send, additional monitoring, bond-based dispute mechanism and Schelling game based distributed judges to increase the robustness of Preddy's security model.**

Keywords—blockchain, Ethereum, smart contracts, tokens, automatic market maker, AMM, prediction market

I. Introduction

Blockchain technologies can be described as a growing list of transactions that has been verified and cannot be erased, maintained across multiple nodes in a peer-to-peer network. [1] Using cryptographic proof instead of a single source of authority, it preserves its trustless property. [2] The decentralized, immutable and trustless nature of blockchain makes its applications promising, revolutionizing the finance industry.

Blockchain has also seen widespread adoption beyond Bitcoin since the inception of Ethereum. [3] With smart contracts, Blockchain can now be utilized for various services, as long as they can be distilled into digital transactions.

Prediction markets are "markets where participants trade in contracts whose payoff depends on unknown future events". [4] These markets have existed since before the 16th century [5], although modern electronic equivalents started appearing near the end of the 1980s. Early variants took the form of simple betting markets, and the economic theory behind their value as predictions was formalised in 1945. [6] They have been used to predict the outcomes of future events, and have shown to be more accurate than expert opinions in events ranging from elections, award show winners and sales numbers. [7] Blockchain provides a betting avenue for prediction markets that is decentralized, paving the way for new blockchain prediction markets (BPMs) like Augur, Stox and Gnosis.

Centralized prediction markets suffer from being closed, constrained, highly censored and hence limit the amount of liquidity in the market. However, the decentralized nature of BPMs lowers the cost to participate, while making it almost frictionless to create new markets. This opens up the market, encouraging the flow of liquidity and hence increasing the accuracy of the predictions. [8]

We have developed a dApp, called Preddy, that is a BPM built over Ethereum. Preddy uses an Automatic Market Maker, modeled after the Balancer Pool outlined in the balancer whitepaper. [9] Preddy allows for more than 2 outcomes in each market and provides real time information regarding potential winnings for each outcome. The most unique feature of Preddy is the custom distribution of initial liquidity on creation of the market - this feature is not available in other BPMs.

BPMs however suffer from issues in security, privacy and scalability due to the underlying blockchain it relies on, the smart contracts that contain it, as well as the cryptoeconomics that comes with the design of a prediction market. Current BPMs have published on multiple smart contract vulnerabilities, and also implemented elaborate dispute policies to counteract bad actors manipulating the market.

In this paper, we detail the security vulnerabilities of Preddy in the contract, network and application layers. For contract code, we will examine common sources of vulnerabilities and present possible preventative measures. For network vulnerabilities, we will discuss new attack vectors made possible by the decentralised nature of Preddy, and suggest possible mitigations. Within the application layer of Preddy, we will explain the limitations of the current market resolution model, and provide potential mechanisms to prevent outcome manipulation. Our novel contribution will be the comprehensive analysis of security threats in decentralised prediction markets, as well as a combination of recommendations to make a robust and secure BPM implementation that correctly incentivises all parties to behave honestly.

II. Motivation and Literature Survey

A. Motivation

Scalability issues in BPMs mostly arise from the underlying blockchain used and are not application specific. Preddy's market maker, inspired by Balancer, uses constant time operations that are independent of the number of bets made per market. The memory requirements of storing user to bet mappings scale linearly with users, but since each user allocates the same amount of memory and thus pays similar gas costs when betting, this is not a concern from a usability perspective. Therefore, while there are potential research areas in BPM scalability, these are not critical to the functionality of Preddy.

Privacy issues arise from external account addresses being traced back to a common identity. A lack of privacy features may lead to a loss in potential participants due to privacy concerns and inaccurate outcomes predictions as participants do not vote independently. However, the impact of these concerns are limited. Ethereum already provides a pseudo anonymous identity as each external account address is randomly generated. Participants may also mask their identities through external mixing services such as Tornado Cash. [10] This provides an acceptable level of privacy which reduces participants' unwillingness to participate. With the above privacy features, it will be computationally expensive to perform network analysis to link addresses to an identity as well. As such, privacy no longer becomes a main issue in BPMs.

Security is the issue that most concerns Preddy. BPM's core feature is the ability of the smart contract to act as an escrow for each market. The accumulated value of all bets per market can add up to very large amounts and stored for weeks, making the contract a very attractive and lucrative target for attacks. Similar dApps with long-term stored value functionality such as The DAO and Parity were successfully hacked even though they were high profile projects [11] , making it evident that security is paramount to Preddy.

Furthermore, our paper is also concerned about the security of Preddy against outcome manipulation attacks. A central tenet of prediction markets is the incentive for all participants to vote according to their own predictions. This is only possible if participants believe that the markets will resolve coherently and according to reality. If certain parties are able to manipulate the resolution process, the incentives would change to predict what the manipulation will be. This is a critical failure mode and would make BPMs unreliable for predictions, strengthening our argument that security is the most important consideration for Preddy.

B. Literature Survey

Decentralised prediction markets have several popular implementations that have already been developed, such as Gnosis, Augur and Stox. In general, they allow any user to:

1. Create markets for a future event
2. Place bets on outcomes
3. Participate in the resolution process
4. Withdraw their earnings

In contrast to centralised prediction markets, these decentralised variants democratise the process and allow any user to participate in all of the above steps. However, this creates some unique challenges that have to be solved, such as ensuring that markets created have coherent questions and outcomes, and the decentralised resolution process is fair. Additionally, since the central authority is replaced by smart contracts where source code is public and immutable, extra care for security must be taken since rollbacks are unlikely to be possible and feasible.

Augur and Gnosis addresses the smart contract security problem through open-source code, a full security audit and a bug bounty program. [12], [13] The smart contracts for both projects are released on GitHub with good documentation and active support from the development team. Developers may raise security issues in the repository or through a discord channel. The two projects also have a robust bug bounty program. Augur rewards up to $25,000 and Gnosis rewards go up to $50,000. Augur and Gnosis also publicly released their security audit results. The audit document details the vulnerabilities with references to actual code and explains how the project has rectified the problems.

Augur solves the outcome manipulation problem by implementing a bond-based progressive report-dispute system that can eventually resolve into a forked universe. [14] Gnosis starts off with a similar mechanism, but the ultimate resolution is delegated to Kleros, [15] a distributed judge. Stox allows the use of either a bond system (without possibility of forks) or a random polling of token holders as the dispute mechanism, prioritising speed of resolution over robustness of disputes. [16] A central theme is the presence of a "fast-path" when outcomes are not disputed, and incentivising oracles to be honest using the threat of dispute.

Froberg et. al outlines the following 5 requirements as what the optimal prediction market has to adhere to: (A) many actors have to be able to participate in the market, (B) no actors are prevented from participating, (C) there is a trustworthy settling function to determine the outcome of event, (D) actors in the market are free to create new contracts at any time and (E) transparency. [7] All following discussions on achieving security in Preddy will be done with these requirements in mind.

III. Observations and Analysis

There are 3 main concerns regarding the security of Preddy - smart contract vulnerabilities, front-running and network attacks as well as the naivety of the dispute mechanism. These correspond to security issues in the contract layer, network layer and application layer respectively.

A. Unidentified Smart Contract Vulnerabilities

Smart contracts are more susceptible to security vulnerabilities as compared to other computer programs [17] as technology is still at an early stage and is not time tested. Furthermore, there is a lack of well defined best practices for programming and testing. Erroneous smart contracts cannot be patched easily due to the immutability of smart contracts as well. Hence, identifying vulnerabilities is critical yet challenging.

Preddy manages the known security vulnerabilities such as the Reentrancy problem within the token withdrawal function.

Term Paper submitted for CE/CZ4153 Blockchain Technology course, NTU.

This is by modifying the internal state before external calls. There are two other security problems that are identified but currently unhandled: (1) Integer Overflow/Underflow and (2) Impossible Withdrawal.

1. Integer Overflow/Underflow occurs when integer operations are performed on variables. Preddy's contracts currently perform all arithmetic operations without checks for overflow. This can be solved by using smart contract libraries that handle overflows such as SafeMath. [18]
2. Impossible Withdrawal is a vulnerability specific to Preddy's design. In Preddy's token withdrawal function, *msg.sender.transfer()* is used which has a 2300 gas limit. If the user has an expensive fallback function or if gas costs changes, the user will be unable to withdraw the funds. A solution will be to use *msg.sender.call.value()* to forward all available gas. [19] However, this can be prone to reentrancy attacks which should also be handled.

There may also be other undiscovered security issues with regards to access control or denial of service. Uncategorised security issues may also arise that are specific to Preddy's design. The main difficulty in smart contract security lies in managing these unknown vulnerabilities which may arise due to lack of knowledge or human oversight. Known security issues can always be fixed easily but might lead to new security loopholes as well. As such, there is a need for frameworks or workflows to systematically identify security issues.

B. Possibility of Front-Running and Network Attacks

Frontrunning and network attacks are seen in every blockchain application. However, they manifest in unique ways in Preddy.

Frontrunning attacks occur because of the process a transaction has to go through between being broadcasted and being published onto the chain. Once a transaction is broadcasted to the network, it will be visible to all the nodes that it is propagated to. If one of these nodes is malicious, it can use the information of this transaction to create its own malicious request, and possibly set a higher gas price with this transaction so that it will eventually be published to the chain first.

In the context of Preddy, the motivation for a malicious node to execute a front running attack is strong. Preddy's betting mechanism rewards people bet on more uncertain markets by assigning more shares for the same price. Hence, in some cases, Preddy will reward earlier betters. Front-running a bet of this nature would give the malicious node more shares for the same price.

Frontrunning attacks are also prevalent in established BPMs. In Gnosis, any bet can be frontrunned because all transaction details will be available in the memory pool of the Ethereum network. [20] It has been acknowledged that this is an issue, which will be fixed in future iterations of Gnosis.

Network attacks also have profound effects on the accuracy of the information Preddy presents. Preddy prides itself on its feature of providing real time price, odds and winnings information. However, these values are calculated based on the node's view of the chain, and relies heavily on the assumption that each node will have the most accurate and most updated view of the chain at all times. While this assumption is reasonable in normal circumstances, a network attack breaks this assumption and heavily compromises the integrity of the information Preddy presents.

In the event of an eclipse attack, the attacker partitions miners so that they can't build on each other's blocks. This gives the attacker the opportunity to double spend, and also create two disparate chains with wrong information. [21] With a more probable and less critical attack like a delay attack, the node is still able to deny affected nodes accurate information. It has been shown that intercepting only 50% of connections to a node can keep the node uninformed for 63% of its uptime. [22]

In a prediction market, it's important for punters to understand the odds of a potential bet. This allows them to make an informed and rational decision by combining that information with their own intuition on the outcome probabilities. However, if Preddy provides inaccurate price/odds discovery, the bets of the punters will change and hence break the integrity of the prediction market. The prediction market can no longer provide accurate predictions.

C. Naivety of Arbitration/Dispute Mechanism

As a decentralised prediction market, Preddy is susceptible to the same problems of agents manipulating the outcome as other prior work, such as in Stox, Gnosis, and Augur. [14] This is known as *outcome manipulation*, and is one of the core issues to be solved for any decentralised prediction market. As a proof-of-concept, Preddy utilises a naive resolution mechanism, where an ethereum address is denoted as the *arbiter* upon market creation, and has complete control of resolving the market to an outcome upon market close. A fee proportional to the market is also given to the arbiter to offset the gas cost, as well as to incentivise timely resolution.

In such a model, it is easy to see the many conflicts of interests that an arbiter can have to manipulate outcomes. A simple but extremely profitable strategy could be to always bet on the underdog and resolve in favour of the arbiter's own bets. Rational punters would then have no incentive to bet on the markets, since they would be rigged and not linked to the actual outcomes in the real world.

Another issue with Preddy's resolution mechanism is the choice of arbiter. The arbiter is chosen at market creation by the market creator, meaning that there is an obvious incentive for the creator to choose an address that they control, or at least an address controlled by an agent that they have influence over. They can then resolve the outcome in their favour and reap the arbiter fee. Since there is no concept of identity in Ethereum, there is no way to know and prevent this from happening.

Furthermore, all of the bets are locked up behind a single address, which enables another issue. The individual controlling

the arbiter address could become incapacitated, lose the private key, or otherwise be unable to resolve the market, and all of the ether would become locked and unredeemable. This critical failure mode, while unlikely due to economic incentives, still presents a challenge to the trustworthiness of Preddy's prediction markets.

All these issues mean that Preddy's attractiveness to potential participants is greatly reduced, and it is unlikely for markets to become popular and attract bets.

IV. Proposed Solutions

In this section, propose potential solutions to address the issues that you found in your analysis earlier.

These solutions may be inspired from the lectures, invited talks, related works, or any other instance of similar development projects.

A. Automated Security Analysis Framework

There are numerous ways to limit unidentified smart contract security vulnerabilities. The primary way is to train developers. Developers will have to learn security attacks to avoid including them in the code. This can be achieved with good design patterns such as "Favor pull over push for external calls" and "Don't delegate call to untrusted code". [19] A deep understanding of the EVM or Solidity's implementation will also help to reduce unexpected code behavior. In addition, developers will need to stay up to date with security trends through conferences or research journals. Hence, this requires a huge investment in human capital.

Alternatively, firms can also employ security experts to conduct a full security audit. However, this is financially costly and time intensive. Another option could be a bug bounty program to utilize the open source community. The effectiveness will depend on the attractiveness of the bug bounty program's reward structure in comparison to the black market's rewards. In summary, these methods are financially costly, time intensive and can be prone to human error. As such, automated security analysis tools are promising alternatives. These open-sourced tools are free, easy to incorporate into the development workflow and offer instant feedback.

There are two main types of security analysis methods which are static analysis and dynamic analysis.

Static analysis involves analysis of the smart contract without executing it. This includes methods such as symbolic execution, pattern recognition and formal verification. [23] An open-source state of the art static analysis tool is Slither. [24] Slither first recovers critical data about the code such as the contract's inheritance graph and control flow graph. It then converts the code to an internal representation language SlithIR to perform code analysis. The code analysis includes the following component:

1. Read/Write, which identifies the read and write operations for all variables.
2. Protected functions, which models the user access control design for each function in the smart contract.
3. Data dependency analysis, which identifies the inter-dependencies of all variables. It also highlights *tainted* variables which are variables that rely on a user-controlled variable.

The information from this pipeline is then exposed through an API. Developers can build custom vulnerability detectors through this API. Slither's built-in detectors include detection of reentrancy bugs, unintialized state/storage variables and shadowing of variables.

Dynamic analysis is a method that analyses a smart contract's behavior during run time. It can be used with static analysis to reduce false positive such as in MAIAN. [25] Dynamic analysis systems usually consist of an input generator and a test oracle. The input generator generates inputs for interaction with the contract while the test oracle monitors contract and transaction properties to ensure correct behaviour. A recent work in this area is ContraMaster. [26] ContraMaster assumes that smart contracts maintain an internal state of users' account balances which will be called bookkeeping balances. This leads to two important invariants which the test oracle will monitor. The balance invariant requires that "the difference between the contract balance and the sum of all participants' bookkeeping balances remains constant, before and after a transaction". This ensures that the bookkeeping balances are maintained properly. The transaction invariant "requires that the amount deducted from a contract's bookkeeping balances is always deposited into the recipient's balance". This ensures that a transaction occurs successfully and that transaction failure is managed by the contract. ContraMaster then generates fuzzing inputs in a targeted way to discover security vulnerabilities. This is done by observing contract states and data dependency to increase path coverage and move towards the root of vulnerabilities. The input generator includes mutations of the function inputs, gas limit, attack contract's fallback function, attacker's transaction sequence and targeted contract's internal state. The transaction sequence is then verified against the test oracle.

A practical usage of these automated security analysis tools is to integrate them into the smart contract development pipeline. There could be an automated workflow where security tests are executed when code is pushed to the repository. This is highly feasible as the tools have a marginal impact on execution time with Slither taking only 5 seconds to analyze a contract. [17] The results of the tests can be then reflected in the code review for developers to take action. A combination of both Slither and ContraMaster can also be used to reduce the risks of false negatives. False negatives are more costly than false positives as contracts cannot be patched in the same way as computer programs.

Despite the benefits of automated security analysis tools, recent research has shown that state of the art tools are only able to accurately detect some types of vulnerability. Furthermore, the most accurate tool only detects 27% of vulnerabilities with a large number of false positives. [17] Hence, a multi-prong approach is still necessary to identify security vulnerabilities and the tools may only be used to aid development. A complete security model will include smart contract security training and

security analysis tools in the development phase, and a full security audit and bug bounty program in the deployment phase.

B. Submarine Send

To prevent frontrunning attacks, submarine send can be used. In submarine send, the transaction is first concealed as an indistinguishable contract, and only after it is mined is it published normally in the blockchain. Preddy can make use of LibSubmarine, an open-source smart contract library, to support submarine sends. [27] The Gnosis protocol, which also suffers from vulnerability to front-running attack, also recommends the two-phase commit and reveal submission as a protection to these attacks.

C. Additional Monitoring

While a lot of the solutions to preventing network attacks require changes to the ethereum network, additional monitoring in each node taking part in the prediction market can effectively prevent and detect adversarial attacks from neighbouring nodes. Enabling these monitoring capabilities cannot be implemented as part of the dApp (as the application will not have control over the network layer), but can serve as recommendations for users of Preddy. As such, we suggest the following recommendations for all user of Preddy to prevent network attacks:

- Monitor round-trip time (RTT). Sudden changes (especially longer round trip times) usually indicates an adversarial attack.
- Monitor distribution of connections
- Monitor time elapsed between requests. If it is longer, it might indicate an adversary delaying the broadcast of the transaction.

If any anomalies arise during monitoring, more random connections should be created to prevent an eclipse attack.

D. Bond-based Dispute Mechanism

A potential solution to the outcome manipulation problem is Augur's dispute mechanism. [14] Similar to Preddy, Augur's markets have a designated reporter chosen at market creation by the market creator. However, designated reporters have a time limit for reporting, after which an open reporting window begins where anyone can report on the outcome. This solves the failure mode where the arbiter is unable or unwilling to report on the outcome.

After an initial report is made and a tentative outcome is chosen, the market enters a dispute round, where reporters are able to dispute the outcome in favour of what they believe to be the truth. To do so, they post a bond with real world monetary value. Once another outcome reaches a certain threshold bond amount, the market is successfully disputed. The bond amounts are carefully chosen so that all reporters have a fixed ROI, making crowdsourcing possible. Through the dispute round, we solve the problem of outcome manipulation by a single arbiter, as a dishonest arbiter would lose their bond.

However, another possibility for malfeasance exists, where a dishonest player with enough financial firepower could overwhelm the dispute process. To mitigate this, the dispute mechanism has a built in feature to slow down each round as the bonds grow bigger, to allow for crowdfunding. Furthermore, when bond amounts grow beyond a certain threshold (set as a percentage of token supply), a mechanism known as a *fork* is initiated, where all markets and token holders have to choose between universes denoted by the outcome. Even if a rich malicious agent were to trigger this fork and influence the eventual resolution into his desired universe, honest agents would not want to participate in such a universe, and the tokens would lose their value. This disincentivises such an attack from happening.

To implement such a system in Preddy, it is necessary to create a separate token for bond posting, since the ability to fork the universe is the crux of the incentive. Afterward, the mechanism can be implemented as an extension of Preddy's smart contract code.

E. Schelling Games as Distributed Dispute Judges

Another way to prevent outcome manipulation is through the use of distributed judges, such as the Kleros protocol. This is used by prediction markets on Gnosis. While the resolution mechanism starts off in a similar fashion to Augur on Realit.io, where increasingly large bonds are posted to dispute outcomes, the eventual outcome of disputes is not a fork, but rather an arbitration mechanism powered by Kleros. [28]

Kleros is a decentralised judge mechanism based on Schelling Game Theory. [29] Potential jury members stake some tokens for a chance to participate in a judging round, which mitigates sybil attacks. A jury panel is chosen randomly from this process, and each member must cast a vote to resolve the outcome. The majority vote wins, and minority voters lose their stake.

Since jury members are unlikely to know each other, and even if a member announces their vote, it is difficult to truly discern what they are really voting for, the game-theoretic optimal behavior of juries is to select based on their belief of truth. This is because such a choice has the highest likelihood of being the majority, a concept known as the Schelling Point. The decision of any Kleros court can also be appealed, by posting a larger arbitration fee, leading to another round of arbitration with double the number of jurors.

Kleros can be integrated into Preddy's resolution mechanism using a similar method to Gnosis, by posting outcomes on Realit.io and allowing Kleros courts to be the final arbiters.

V. Conclusion

In this paper, we have outlined the following security issues in Preddy as well as recommendations to counteract it.

TABLE I. Summary of Security Issues and Recommendations for Preddy

Layer	Security Issues	Recommendations
Contract	Smart contract vulnerabilities	automated security analysis framework
Network	front-running and network attacks	submarine send
		additional monitoring user recommendations
Application	naivety of the dispute mechanism	bond-based dispute mechanism
		schelling games

In all cases, the security recommendations ensure that they do not compromise the 5 requirements of a prediction market outlined by Froberg et.al - (A) many actors have to be able to participate in the market, (B) no actors are prevented from participating, (C) there is a trustworthy settling function to determine the outcome of event, (D) actors in the market are free to create new contracts at any time and (E) transparency. These recommendations hence will make Preddy a more secure platform, while maintaining its key features as a decentralized prediction market.

REFERENCES

[1] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "BlockChain Technology: Beyond Bitcoin," *Appl. Innov. Rev.*, no. 2, 2016.

[2] S. Nak moto, "Bitcoin: A Peer-to-Peer Electronic Cash System."

[3] "What is Ethereum? | ethereum," *ethereum*. https://ethereum.org/en/what-is-ethereum/.

[4] J. Wolfers and E. Zitzewitz, "Prediction markets," *J. Econ. Perspect.*, vol. 18, no. 2, pp. 107–126, 2004, doi: 10.1257/0895330041371321.

[5] P. Rhode and K. Strumpf, "Historical Political Futures Markets: An International Perspective," Cambridge, MA, Oct. 2008. doi: 10.3386/w14377.

[6] "Biography of Ludwig Edler von Mises (1881–1973)," *The Library of Economics and Liberty*. https://www.econlib.org/library/Enc/bios/Mises.html.

[7] E. Fröberg, G. Ingre, and S. Knudsen, "Blockchain and prediction markets," 2019, [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-249988.

[8] "The Ultimate Guide to Decentralized Prediction Markets," *Augur*, 2018. https://augur.net/blog/prediction-markets.

[9] F. Martinelli and N. Mushegian, "Balancer Whitepaper," 2019. https://balancer.finance/whitepaper/.

[10] "Tornado.cash," *Tornado.cash*. .

[11] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart Contract: Attacks and Protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020, doi: 10.1109/ACCESS.2020.2970495.

[12] "Augur Bug Bounty Program," *Augur*. https://www.augur.net/developers/.

[13] "Code and Security - Gnosis Developer Portal," *Gnosis*. https://docs.gnosis.io/protocol/docs/devguide04/.

[14] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform," Jan. 2015, doi: 10.13140/2.1.1431.4563.

[15] S. JAMES, "A Good Omen - Kleros x Gnosis x DxDAO and Realitio Align with Conditional," *Kleros*, 2020. https://blog.kleros.io/a-good-omen-kleros-x-gnosis-x-dxdao-align-with-conditional/.

[16] "Stox Platform for Prediction Markets."

[17] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical review of automated analysis tools on 47,587 Ethereum smart contracts," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, Jun. 2020, pp. 530–541, doi: 10.1145/3377811.3380364.

[18] "Contracts - OpenZeppelin Documentation," *OpenZeppelin*. https://docs.openzeppelin.com/contracts/2.x/.

[19] "Ethereum Smart Contract Security Best Practices," *ConsenSys Diligence*. https://consensys.github.io/smart-contract-best-practices/.

[20] "Smart Contracts - Gnosis Developer Portal," *Gnosis*, 2020. https://docs.gnosis.io/protocol/docs/devguide03/#solution-submission-is-front-runnable.

[21] E. Heilman, A. Kendler, T. Aviv Zohar, and S. Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network Ethan," *USENIX Secur.*, 2015.

[22] V. K. Gostishchev, N. V. Stal'tsev, L. F. Muliaev, and A. G. Khanin, "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies," *Vestn. Khir. Im. I. I. Grek.*, vol. 136, no. 1, pp. 69–70, 1986.

[23] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, "Security Analysis Methods on Ethereum Smart Contract Vulnerabilities: A Survey," Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.08605.

[24] J. Feist, G. Grieco, and A. Groce, "Slither: A Static Analysis Framework for Smart Contracts," in *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, May 2019, pp. 8–15, doi: 10.1109/WETSEB.2019.00008.

[25] I. Nikolic, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding The Greedy, Prodigal, and Suicidal Contracts at Scale," Feb. 2018, [Online]. Available: http://arxiv.org/abs/1802.06038.

[26] H. Wang, Y. Li, S.-W. Lin, C. Artho, L. Ma, and Y. Liu, "Oracle-Supported Dynamic Exploit Generation for Smart Contracts," Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.06605.

[27] and S. E. Lorenz Breidenbach, Tyler Kell, Stephane Gosselin, "LibSumbarine." [Online]. Available: https://libsubmarine.org/.

[28] L. Paper, W. George, and F. Ast, "Kleros Previous Work : SchellingCoin Mechanism," no. March, pp. 1–45, 2020, [Online]. Available: https://www.allcryptowhitepapers.com/kleros-whitepaper/.

[29] T. C. Schelling, "Prospectus for a Reorientation of Game Theory," *Rand*, 1959.