Trusting a Smart Contract Means Trusting Its Owners: Understanding Centralization Risk

1st Metin Lamby
Technical University of Munich
Munich, Germany
metin.lamby@tum.de

2nd Valentin Zieglmeier
Technical University of Munich
Munich, Germany
valentin.zieglmeier@tum.de

3rd Christian Ziegler
Technical University of Munich
Munich, Germany
christian.ziegler@tum.de

***Abstract*—Smart contract access control mechanisms can introduce centralization into supposedly decentralized ecosystems. In our view, such centralization is an overlooked risk of smart contracts that underlies well-known smart contract security incidents. Critically, mitigating the known vulnerability of missing permission verification by implementing authorization patterns can in turn introduce centralization. To delineate the issue, we define centralization risk and describe smart contract source code patterns for Ethereum and Algorand that can introduce it to smart contracts. We explain under which circumstances the centralization can be exploited. Finally, we discuss implications of centralization risk for different smart contract stakeholders.**

***Index Terms*—Smart contracts, Vulnerabilities, Risks, Access control, Centralization, Trust**

I. Introduction

One core goal of ecosystems leveraging permissionless blockchain technology is decentralization. However, studies on major blockchain systems indicate that this property is currently not fully achieved [e.g., 1]. For example, in Proof of Work (PoW) blockchains, the majority of mining resources are concentrated in a small number of nodes or mining pools [2]. Centralization also occurs in governance, for example in the decision of core developers to lower the minimum transaction fee in Bitcoin [3]. Similarly, decentralized finance (DeFi), specifically smart contracts, promises to be of use when conducting transactions without having to rely on a central entity within the ecosystem. Smart contracts can be leveraged as agreements between mutually distrusting participants. They are automated by the consensus mechanism of the underlying blockchain, removing the role of a trusted authority.

In this paper, however, we argue that smart contracts themselves can be a source of centralization. Privileged authorization to contract functionality is implemented with access control patterns. Even though restricting access appears to be a logical step in an open ecosystem [4], it may harm its decentralized nature. The dilemma between authorization and centralization sources implemented into a smart contract appears to be a question of priorities. As this may impose risks on stakeholders, the dilemma needs to be discussed.

This work was supported by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Algorand Foundation.

Since DeFi is a developing and open ecosystem, it is challenging to protect all stakeholders. Therefore, many security incidents occur, including unexpected financial loss to users, liquidity providers, and other parties [5]. Stakeholders suffered a total loss of at least US$ 3.24 billion after interacting with the DeFi ecosystem from Apr 30, 2018 to Apr 30, 2022 [5]. The most common DeFi incident cause are smart contract vulnerabilities [4], [5], including hacks due to user privileges.

II. Related Work

The code pattern of introducing privileged parties into smart contracts is coined as a "tainted owner variable" vulnerability by Brent, Grech, Lagouvardos, *et al.* [6] and an "overpowered role" vulnerability in the GitHub repository of Tikhomirov, Voskresenskaya, Ivanitskiy, *et al.* [7] which is part of the publication itself. The issue exists if a function is callable by a single privileged user, resulting in the underlying contract having a dependency on the respective user address. Undesirable consequences for investors may occur if the private key of the superuser address becomes compromised [8].

To detect such issues in Ethereum ERC token contracts, Ma, Ren, Ouyang, *et al.* [9] provide an application that leverages a hybrid analysis method integrating datalog analysis and directed fuzzing. Furthermore, Fröwis and Böhme [10] use symbolic execution to reveal ownership structures in any given Ethereum smart contract.

III. Understanding Centralization Risk

In the following, we first define centralization risk. Then, we outline how access controls can create and exacerbate it. Finally, we consider the relevance for research and practice. We explain the importance of centralization risk in practice and highlight an identified awareness gap in academia.

A. Definition

We define centralization-related risks in smart contracts based on academic research and the practitioner's perspective. To uncover relevant sources, we used results from both a systematic and an exploratory literature review. In academia, Ma, Ren, Ouyang, *et al.* denote centralization risk as "backdoor threats" and define them as "threats related to high-privileged functions" [9, pp. 1–2]. On the practicioner's side, the smart contract auditor *Certik* classifies the vulnerability with major

severity and defines findings related to centralization as application logic that acts against the nature of decentralization, including "specialized access roles in combination with a mechanism to relocate funds" [11]. Finally, *Coinbase* defines the issue as "superuser risks" and flags their occurrence when "single actors have the sole authority to execute a dangerous function" [12]. We can therefore define centralization risk in smart contracts as the source code including privileged access patterns on fund-modifying logic.

B. Access Controls Becoming Risks

The "access control" vulnerability, caused by inadequate authentication enforced by a contract on critical contract functionality [4], conflicts with centralization risk. Implementing access control mechanisms *introduces* user privileges into smart contracts. Therefore, they are an important factor that creates sources of centralization. This can implicate risks, especially when private keys of privileged users are leaked [4]. If the private key of a privileged user's address is compromised, an attacker can take control of critical contract functionality, triggering undesirable actions for stakeholders. An example of a potential attack goal is the artificial dilution or inflation of the value of a token implemented as a smart contract [6]. Therefore, negligent private key management can make authorization patterns vulnerable even though they might initially be implemented to reduce contract risks. For example, consider the Ronin hack: In March 2022, Axie Infinity's Ronin Network experienced a hack that resulted in the theft of US$ 625 million. The attackers compromised a privileged user's private keys and conducted restricted withdrawals on the Ronin bridge. This lead to the Axie DAO validator signature to be abused, resulting in the theft of funds. The following network disruption affected both the Ronin Bridge and the Katana automated market marker [13].

Because of the interplay between authorization patterns and private keys granting access, we argue that centralization risk are multidimensional, consisting of an implementation and a social component. Subsequently, authorization concepts should be evaluated carefully before contract deployment. Yet, if avoided, other vulnerabilities might emerge, including the risk of making applications self-destructible [6]. Which risk to accept becomes a question of priorities as we further discuss below.

C. Relevance in Practice

Major players from the practitioners' community highlight the importance of centralization risk. Their caution is warranted, as centralization issues were the most common attack vector exploited in 2021 [14]. According to the report, US$ 1.3 billion in user funds were lost across 44 DeFi hacks due to user privileges in 2021.

D. Awareness Gap in Academia

Even though the practical relevance of centralization risk is clear, we find that there is an awareness gap in academia. The vulnerability is not mentioned in any academic paper on Algorand smart contract vulnerabilities that we screened in an exploratory review of the field. When reviewing the literature on Ethereum smart contract vulnerabilities, we only found two papers that mention it, namely [6], [9]. The lack of academic research on this vulnerability suggests low relevance. However, as we note above, the risks are of significant importance in practice. Therefore, we find a delta in the suggested relevance of centralization risk between research and practice. This suggests a lack of attention in academia.

IV. Detecting Centralization Risk

Static analysis can be leveraged to detect patterns in smart contracts that contribute to a potential centralization risk-related vulnerability. In the following, we show detection patterns for Ethereum (Solidity) and Algorand (TEAL).

A. Solidity Patterns and Slither Detector

There are different Solidity source code patterns with which *access control* to contract functions can be implemented, as illustrated in table I. Regardless of the pattern used, Solidity access control implementations usually check whether the global variable `msg.sender`, which allows contract developers to read the address of the sender of the current contract call, satisfies certain conditions [15]. In addition to access control patterns, there is one common Solidity source code pattern which is often used in smart contracts to *relocate funds*.

TABLE I
Solidity centralization risk related vulnerability patterns

Pattern	Code Example

onlyOwner modifier [6], [9], [16], [17]	`modifier only_owner {`
	`  require(msg.sender == address(...));`
	`  ...`
	`}`
	
	`function fun(...) only_owner ... { ... }`

require within function [15], [17], [18]	`function fun() ... {`
	`  require(address(...) == msg.sender)`
	`  ...`
	`}`

if statement within function [17]	`function fun() ... {`
	`  if (msg.sender == address(...)) {`
	`  ...`
	`  }`
	`}`

balance modification [19], [20]	`mapping(address => uint) bals;`
	
	`function fun (...) ... {`
	`  bals[...] = bals[...].add(...);`
	`}`

B. TEAL Patterns and Tealer Detector

In order to determine how to detect centralization in TEAL smart contracts, we take advantage of non-academic sources as well as compilation of PyTEAL patterns. First, the Algorand GitHub[1] provides TEAL contracts that we can use to identify access control and balance modification patterns in TEAL. In addition, Matteo [21] implements an authentication logic in PyTEAL. From those, we derive the patterns in II.

TABLE II
TEAL CENTRALIZATION RISK RELATED VULNERABILITY PATTERNS

Pattern	Code Example

address comparison with assert [21], [22]	`byte "manager"`
	`app_global_get`
	`txn Sender`
	`==`
	`assert`

address comparison with branch opcodes [23], [24]	`byte "Creator"`
	`app_global_get`
	`txn Sender`
	`==`
	`...`
	`bz failed`

balance modification [23]	`...`
	`byte "MyBalance"`
	`...`
	`app_local_put`

V. IMPLICATIONS

There are multiple implications that arise from this risk, affecting investors, developers, but also CEXs.

A. Trust Assumptions for Investors

Even though smart contract code is automatically run by the consensus mechanism of a blockchain without a trusted authority, the necessity for trust remains. DeFi aims to eliminate single points of attack that exist in the traditional financial ecosystem [25], but they can be reintroduced by access control patterns on critical smart contract functionality. Therefore, while the enforcement of business logic might be distributed, stakeholders could depend on trusted authorities to execute contract functionality in their favor. As a result, investors engaging in tokenized assets may be exposed to trust assumptions [26] that are difficult to evaluate. If exploited, the vulnerability can cause token holders to incur a loss on their investment. While the reliable parties in CeFi are the intermediaries, privileged access to critical smart contract functionality creates a dependency on a small set of accounts within the DeFi building blocks. This seems to be a questionable ideal as the notion of decentralization loses validity.

[1]Algorand GitHub: https://github.com/algorand/smart-contracts

B. Implementation Dilemma for Developers

Contract developers are exposed to the dilemma of implementing access control patterns and introducing overpowered user roles. Whether to restrict a function becomes a question of priorities affecting many stakeholders. In addition, the concept of multiple signature addresses may gain relevance for contract developers. Multisignature (multisig) wallets refer to a "type of account that requires a minimum number of addresses to sign a transaction before executing it" [26]. Using a multisig for access control introduces an extra layer of security, since operations on the underlying contract require consent from multiple parties [27]. This makes it more difficult for a malicious user to manipulate sensitive contract functions, whether the user is the owner of a contract or is assigned a specific role. Hence, decentralization is increased, as no single entity has total control of a contract function [26]. This technique divides responsibility for key management between multiple parties and prevents the loss of a single private key from potentially causing undesirable and irreversible consequences [27]. Nevertheless, the multisig concept reduces centralization risk but does not mitigate the problem as private key leakages of multiple privileged users have caused smart contract exploits in the past. Also, the mentioned benefits come with the cost of worse usability and increased reaction time when signing a transaction.

C. Increased Security Challenges for CEXs

Further affected stakeholders of this discussion might be CEXs that enable the trade of tokens, their affiliated custody providers, and smart contract auditors. CEXs implement listing processes that aim to identify vulnerable token contracts to prevent them from being listed. Usually, exchanges hire third-party token audits to provide that service. Often, auditors leverage automated smart contract vulnerability detection tools to increase workflow efficiency. Nevertheless, the social component of centralization risk makes it impossible for auditors to detect the vulnerability automatically. Detection tools can only determine the existence of implementation patterns that *may cause* the vulnerability. However, since contracts containing superuser roles on critical functions are said to be as secure as the protections on those roles [12], they need to be evaluated manually. Therefore, smart contracts may not only need to be audited on a source code level but also on a social level, represented by the roles that have access to restricted function execution. A know your customer (KYC) process including an audit on private key management might be an initiative to evaluate the social risk vector a smart contract possesses. As a result, CEXs might introduce additional burdens for token listings while prioritizing those parties admitting their tokens for listing that have a good reputation, possibly bigger institutions. If the addressed risk is neglected, possible consequences might include regulatory scrutiny and reputational damage.

VI. CONCLUSION

Smart contract access control mechanisms can introduce sources of centralization into supposedly decentralized ecosys-

tems, including DeFi. We observe a delta in the perceived relevance of centralization risk between academia and industry, suggesting a research gap. We address this gap by (1) identifying centralization risk and its characteristics and (2) presenting patterns to detect it with static analysis. Our findings implicate increased trust assumptions for token investors, greater development efforts and responsibility for software engineers, and security challenges for CEXs. However, centralization risk is multidimensional and includes a social component. While source code patterns that enable the vulnerability can be detected automatically, the private key management of authorized users requires manual security audits.

To conclude, the introduction of privileged access to functionality within a smart contract can create single points of attack, leading to undesirable consequences if exploited. Therefore, we argue that trusting a smart contract means trusting its "owners," meaning the privileged stakeholders as well as their key management.

References

[1] A. R. Sai, J. Buckley, B. Fitzgerald, and A. L. Gear, "Taxonomy of centralization in public blockchain systems: A systematic literature review," *Information Processing & Management*, vol. 58, no. 102584, 2021.

[2] A. Beikverdi and JooSeok Song, "Trend of centralization in bitcoin's distributed network," in *Proc. 16th SNPD*, IEEE, 2015, pp. 1–6.

[3] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is bitcoin a decentralized currency?" *IEEE Security & Privacy*, vol. 12, no. 3, pp. 54–60, 2014.

[4] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on Ethereum systems security," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–43, 2021.

[5] L. Zhou, X. Xiong, J. Ernstberger, *et al.*, "SoK: Decentralized finance (DeFi) attacks," 2022. arXiv: 2208.13035.

[6] L. Brent, N. Grech, S. Lagouvardos, B. Scholz, and Y. Smaragdakis, "Ethainter: A smart contract security analyzer for composite vulnerabilities," in *Proc. 41st PLDI*, ACM, 2020, pp. 454–469.

[7] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "Smartcheck," in *Proc. 1st WETSEB*, ACM, 2018, pp. 9–16.

[8] SmartCheck, *Smartcheck solidity overpowered roles*, 2019. [Online]. Available: https://perma.cc/7ZS8-39GL (visited on 2023-05-18).

[9] F. Ma, M. Ren, L. Ouyang, *et al.*, "Pied-Piper: Revealing the backdoor threats in Ethereum ERC token contracts," *ACM Trans. Softw. Eng. Methodol.*, 2022.

[10] M. Fröwis and R. Böhme, "Detecting privileged parties on ethereum," in *Proc. 7th WTSC*, Forthcoming, 2023.

[11] Certik, *Fidelis audit report*. [Online]. Available: https://perma.cc/35R3-7MQ5 (visited on 2023-04-25).

[12] The Coinbase Digital Asset & Protocol Security Team, *How coinbase reviews tokens on Ethereum for technical security risks*, 2022. [Online]. Available: https://tinyurl.com/howcoinbrevws (visited on 2022-12-06).

[13] J. Scharfman, "Decentralized finance (DeFi) fraud and hacks: Part 2," in *The Cryptocurrency and Digital Asset Fraud Casebook*, Springer, 2023, pp. 97–110.

[14] Certik, "The state of DeFi security 2021," Tech. Rep., 2021. [Online]. Available: https://perma.cc/593E-VTJK (visited on 2022-12-06).

[15] Y. Xue, M. Ma, Y. Lin, Y. Sui, J. Ye, and T. Peng, "Cross-contract static analysis for detecting practical reentrancy vulnerabilities in smart contracts," in *Proc. 35th ASE*, ACM, 2020, pp. 1029–1040.

[16] C. Ferreira Torres, M. Baden, R. Norvill, B. B. Fiz Pontiveros, H. Jonker, and S. Mauw, "Ægis: Shielding vulnerable smart contracts against attacks," in *Proc. 15th AsiaCCS*, ACM, 2020, pp. 584–597.

[17] J. Schiffl, M. Grundmann, M. Leinweber, O. Stengele, S. Friebe, and B. Beckert, "Towards correct smart contracts: A case study on formal verification of access control," in *Proc. 26th SACMAT*, ACM, 2021, pp. 125–130.

[18] C. Ferreira Torres, H. Jonker, and R. State, "Elysium: Context-aware bytecode-level patching to automatically heal vulnerable smart contracts," in *Proc. 25th RAID*, ACM, 2022, pp. 115–128.

[19] M. Wohrer and U. Zdun, "Smart contracts: Security patterns in the Ethereum ecosystem and solidity," in *2018 International Workshop on Blockchain Oriented Software Engineering*, IEEE, 2018, pp. 2–8.

[20] J. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," in *Proc. 2019 WETSEB*, IEEE, 2019, pp. 8–15.

[21] B. Matteo, "Decentralized carpooling with algorand blockchain," Ph.D. dissertation, Ca' Foscari Unversity of Venice, 2022.

[22] AfricaCodeAcademy, *Fidelis teal approval program*. [Online]. Available: https://perma.cc/SJ9R-JDMC (visited on 2023-04-28).

[23] Algorand, *Algorand crowd_fund.teal smart contract*. [Online]. Available: https://perma.cc/7LED-BRU8 (visited on 2023-04-28).

[24] Algorand, *Algorand dex.teal smart contract*. [Online]. Available: https://perma.cc/H4YN-2PRH (visited on 2023-04-25).

[25] F. Schär. "Decentralized finance: On blockchain- and smart contract-based financial markets." Available at SSRN: https://ssrn.com/abstract=3843844. (2020).

[26] R. Behnke, *Designing secure access control for smart contracts*, 2022. [Online]. Available: https://perma.cc/58Q2-3K8N (visited on 2023-02-01).

[27] C. Smith *et al.*, *Smart contract security*, 2023. [Online]. Available: https://perma.cc/W9SQ-RKYV (visited on 2023-01-31).