# Automated Analysis of Cardano Smart Contracts

Raghav Kumar*
Aakash Wagle*
rfk5511@psu.edu
aaw5833@psu.edu
The Pennsylvania State University
State College, Pennsylvania, USA

## Abstract

Cardano offers a novel approach to executing smart contracts through its extended UTXO (eUTXO) model, which strikes a balance between Bitcoin's restrictive design and Ethereum's more flexible global state. Our goal is to find code analysis methods that are effective at detecting bugs and anti-patterns in Cardano smart contracts. [1]

## Keywords

Smart Contract Vulnerability, Code Analysis, Cardano

**ACM Reference Format:**
Raghav Kumar and Aakash Wagle. 2025. Automated Analysis of Cardano Smart Contracts. In . ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Problem Statement

In the Cardano blockchain, the global state is represented by the information held in Extended Unspent Transaction Output (eUTXO). The complete eUTXO model is described by Brunjes and Gabbay [7]. Broadly, the global state—which includes the value owned by every user in the network and other data such as asset ownership—can be determined by examining all unspent transaction outputs at any given time.

To alter the global state, one must sign and submit a transaction specifying which unspent transaction outputs will be consumed and which new ones will be created. The data stored in the proposed output UTXO, upon transaction acceptance, represents the new state of the system.

In addition to default sanity checks—such as ensuring a user only consumes UTXOs they own and the output values do not exceed input values—developers can define custom validation functions. These functions, termed as "Validators," return `true` if a UTXO can be used as a valid input in a transaction. Validators have access to:

- A redeemer, which acts as a function argument indicating the operation performed in the transaction.
- Context, which includes data of the UTXO associated with this validator, data of other input UTXOs, and data related to the proposed output UTXOs.

Based on this information, the validator returns `true` if the usage of its token is deemed valid. Unlike Ethereum, where validating functions have complete access to global state, Cardano's validators access limited information, making it a novel system with underexplored security properties.

Validators encapsulate the business logic of assets, and all UTXOs with the same validator function share the same validator hash, allowing off-chain observers to identify tokens linked to specific business logic. Validator bugs, or intentionally used anti-patterns, can allow attackers to bypass the intended or desired business logic, potentially leading to financial losses. Therefore, our objective is to perform a security analysis of Validator functions and assess the effectiveness of various verification methods used in Cardano.

A collection of Validator bugs/anti-patterns is provided by Vacuum Labs in their capture-the-flag contest [11]. In the first phase of our study, we will utilize these challenges, and demonstrate how insecure Validators may allow attackers to gain undue value. The Cardano ecosystem supports validators in multiple languages, including Haskell-based Plutus, Aiken, Python, and Rust. Vacuum Labs' capture-the-flag contest employs Aiken.

In the second phase of our study, our aim is to:

(1) Identify analysis techniques that can detect vulnerabilities due to bugs or anti-patterns in the capture-the-flag challenges.
(2) Determine the most effective representation for analysis—whether the original source code (in Aiken [1], Python [4], Rust [3], TypeScript [5], or Haskell [6]) or its Haskell translation.
(3) Collect source code of Validators deployed on the Cardano network, or available on Github, clean them and manually analyse to find new anti-patterns or bugs.
(4) Execute the proposed analysis methods on open-source Validator source code available in the Cardano ecosystem [2].

## 2 Motivation

Blockchains enable various emerging use cases where no single entity exercises supreme control. The fundamental principle is to create a system of contracts with pre-defined rules, ensuring that no party can renege on their commitments. However, if the rules are incorrectly specified, the system may validate transactions that should not be approved. Consequently, validating the code that enforces these rules is crucial.

*Both authors contributed equally to this research.
[1]ChatGPT used as LaTex generator and search engine. The written content belongs to the authors.

The first blockchain network, Bitcoin, did not emphasize enabling developers to express customized rules. Ethereum changed this paradigm by introducing smart contracts, which have since become a cornerstone for decentralized systems. Following Ethereum, numerous blockchain platforms have emerged, allowing developers to implement desired logic through smart contracts verified by network participants. Different platforms employ distinct mechanisms for enabling smart contracts, which results in varying attack surfaces.

The smart contracts on the Ethereum blockchain have been extensively analyzed for vulnerabilities [8–10, 12, 13, 15] as well as anti-patterns, and numerous tools have been developed for this purpose. Similarly, a study focusing on the security of Algorand's smart contracts has also been conducted [14]. However, literature addressing the security analysis of Cardano's smart contracts remains sparse. This scarcity might be attributed to the strong guarantees provided by functional programming languages like Haskell, but human errors in source code are always possible.

Furthermore, Cardano's approach to maintaining global state is distinct from Ethereum's, and it represents a significant class of blockchains that use the extended unspent transaction output (eUTXO) model.

## 3 Proposed Methodology

(1) **Setup and Demonstration of Attacks:**
- Set up a local Cardano network and deploy contracts with vulnerable validators.
- Use a web interface to demonstrate how attackers can exploit vulnerabilities in smart contracts.

(2) **Techniques Evaluation:**
- Examine techniques applied to verify the correctness of Ethereum and Algorand smart contracts for their applicability to Cardano smart contracts.
- Explore additional analysis techniques for their suitability.

(3) **Evaluation Sources:**
- *Real-world Contracts:* Collect open source code of Validators by exploring the Cardano network and Github [2].
- *Common Bugs:* Analyze anti-patterns described in [11], including issues like integer overflows/underflows and anti-patterns observed manually in collected Validator source code.
- *Analysis Techniques:* Investigate mechanisms described in [8, 12, 15] and elsewhere for analyzing Validators collected from the web.

## References

[1] 2023. Aiken Documentation. https://docs.cardano.org/developer-resources/smart-contracts/aiken.

[2] 2023. Cardano Open Source Index. https://github.com/micahkendall/cardano-opensource-index.

[3] 2023. Naumachia Rust Library. https://github.com/free-honey/naumachia.

[4] 2023. OpShin Python Documentation. https://developers.cardano.org/docs/smart-contracts/opshin/.

[5] 2023. Plu-ts TypeScript Documentation. https://developers.cardano.org/docs/smart-contracts/plu-ts.

[6] 2023. Plutus Haskell Documentation. https://developers.cardano.org/docs/smart-contracts/plutus.

[7] Lars Brunjes and Murdoch Gabbay. 2020. The Extended UTXO Model. https://arxiv.org/pdf/2003.14271.

[8] Joel Frank, Cornelius Aschermann, and Thorsten Holz. 2020. ETHBMC: A Bounded Model Checker for Smart Contracts. https://www.usenix.org/conference/usenixsecurity20/presentation/frank. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2757–2774.

[9] Fabio Gritti, Nicola Ruaro, Robert McLaughlin, Priyanka Bose, Dipanjan Das, Ilya Grishchenko, Christopher Kruegel, and Giovanni Vigna. 2023. Confusum Contractum: Confused Deputy Vulnerabilities in Ethereum Smart Contracts. https://www.usenix.org/conference/usenixsecurity23/presentation/gritti. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 1793–1810.

[10] Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. 2020. BlockSci: Design and applications of a blockchain analysis platform. https://www.usenix.org/conference/usenixsecurity20/presentation/kalodner. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2721–2738.

[11] Vacuum Labs. 2023. Cardano Capture the Flag. https://github.com/vacuumlabs/cardano-ctf.

[12] Christoph Sendner, Lukas Petzi, Jasper Stang, and Alexandra Dmitrienko. 2023. Vulnerability Scanners for Ethereum Smart Contracts: A Large-Scale Study. https://arxiv.org/abs/2312.16533. arXiv:2312.16533 [cs.CR]

[13] Tianle Sun, Ningyu He, Jiang Xiao, Yinliang Yue, Xiapu Luo, and Haoyu Wang. 2024. All Your Tokens are Belong to Us: Demystifying Address Verification Vulnerabilities in Solidity Smart Contracts. https://www.usenix.org/conference/usenixsecurity24/presentation/sun-tianle. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 3567–3584.

[14] Zhiyuan Sun, Xiapu Luo, and Yinqian Zhang. 2023. Panda: Security Analysis of Algorand Smart Contracts. https://www.usenix.org/conference/usenixsecurity23/presentation/sun. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 1811–1828.

[15] Christof Ferreira Torres, Mathis Steichen, and Radu State. 2019. The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts. https://www.usenix.org/conference/usenixsecurity19/presentation/ferreira. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1591–1607.