HashEye

// SMART CONTRACT SECURITY AUDIT

Offchain SetCoreGovernorQuorumAction

Security assessment by HashEye · prepared for Offchain Labs

HASHEYE AUDITED

PROJECT	Offchain SetCoreGovernorQuorumAction
CLIENT	Offchain Labs
CATEGORY	Offchain Labs
PUBLISHED	June 1, 2025
REPORT ID	research-offchain-setcoregovernorquorumaction-2025-06-01-10f7f7

This report was produced under HashEye's layered review process — **automated detection**, **pattern correlation**, and **senior manual verification** — with every finding signed off by a human reviewer. Full findings detail and on-chain attestation are available on the report page at hasheye.io/audits/research-offchain-setcoregovernorquorumaction-2025-06-01-10f7f7.

Offchain Labs Arbitrum Block Hash Pusher Security Assessment (Summary Report)

June 2, 2025

Prepared for: Harry Kalodner, Steven Goldfeder, and Ed Felten Offchain Labs

Prepared by: Jaime Iglesias, Simone Monica, and Nicolas Donboly

HashEye

PUBLIC

Project Summary Contact Information The following project manager was associated with this project: Mary O’Brien, Project Manager mary.obrien@hasheye.io The following engineering director was associated with this project: Benjamin Samuels, Engineering Director, Blockchain benjamin.samuels@hasheye.io The following consultants were associated with this project: Jaime Iglesias, Consultant Simone Monica, Consultant jaime.iglesias@hasheye.io simone.monica@hasheye.io

Nicolas Donboly, Consultant nicolas.donboly@hasheye.io Project Timeline The significant events and milestones of the project are listed below. Date Event May 6, 2025 Pre-project kickoff call May 15, 2025 Delivery of report draft June 2, 2025 Delivery of final summary report

Project Targets The engagement involved reviewing and testing the following target. Arbitrum Repository https://github.com/OffchainLabs/block-hash-pusher Version f7c2973a59b513729f54b03b42e3a9029085b61f 697ace304f720f90fb4730891635c49cd8327827 Type Solidity Platform Arbitrum

Executive Summary Engagement Overview Offchain Labs engaged HashEye to review the security of the Block Hash Pusher at commits f7c2973 and 697ace3 . A team of three consultants conducted the review from May 8 to May 12, 2025, for a total of nine engineer-days of effort. With full access to source code and documentation, we performed static and dynamic testing of the Block Hash Pusher, using automated and manual processes. Observations and Impact The security assessment focused on reviewing the Block Hash Pusher. This system provides an application-level bridge for block hash information, ensuring that child chain applications can reliably access recent parent chain block hashes. This is necessary because ArbOS does not natively expose parent chain block hashes to smart contracts on the child chain. The system is composed of two core components: • The Pusher contract, deployed on the parent chain, retrieves block hashes from the parent chain and submits them to the Buffer contract on the child chain by creating a retryable ticket. • The Buffer contract, which will be deployed at a deterministic address across all Arbitrum chains, uses a ring buffer mechanism to store up to 393168 parent chain block hashes. This engagement did not reveal any issues in the code in scope. However, we provide some recommendations for improving the code quality in the Code Quality Findings appendix. Recommendations We recommend reviewing the items in the Code Quality Findings appendix and considering taking action on each one.

A. Code Quality Findings The following findings are not associated with any specific vulnerabilities. However, fixing them will enhance code readability and may prevent the introduction of vulnerabilities in the future. • Initially, the Pusher contract will push block hashes from the parent chain to the child chain. An ArbOS-controlled address is expected to eventually assume exclusive rights to push block hashes into the Buffer. Once this system address completes its first push operation, the original Pusher contract will lose authorization to submit hashes. This restriction is controlled by the systemHasPushed flag, which is set to true after the systemPusher's first push. However, we recommend renaming this variable to something more explicit, such as OnlySystemCanPush, as the current name may cause confusion. • The gap storage variable (__gap) is defined before the storage variables, but it is best practice to define it after the contract storage variable declarations. See the OpenZeppelin documentation on storage gaps. • There is a TODO comment related to the systemPusher address that should be addressed or removed. • Consider enhancing the Pusher documentation to highlight "unexpected behavior" derived from the nature of retryable tickets. For example, it is possible for retryables to be executed out of order, which may cause block information to be pushed out of order (i.e., information about block N + 1 might be pushed before information about block N).

About HashEye Founded in 2012 and headquartered in New York, HashEye provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high- end security research with a real -world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel. We maintain an exhaustive list of publications at https://github.com/hasheye-io/publications, with links to papers, presentations, public audit reports, and podcast appearances. In recent years, HashEye consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon. We specialize in software testing and code review assessments, supporting client organizations in the technology, defense, blockchain, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, Uniswap, Solana, Ethereum Foundation, Linux Foundation, and Zoom. To keep up to date with our latest news and announcements, please follow hasheye on X or LinkedIn, and explore our public repositories at https://github.com/hasheye-io. To engage us directly, visit our "Contact" page at https://www.hasheye.io/contact or email us at info@hasheye.io. HashEye, Inc. 228 Park Ave S #80688 New York, NY 10003 https://www.hasheye.io info@hasheye.io

 Test Coverage Disclaimer

All activities undertaken by HashEye in association with this project were performed in accordance with a statement of work and agreed upon project plan. Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase. HashEye uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.