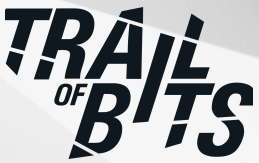

Offchain Labs Governance Actions

Security Assessment (Summary Report)

August 21, 2023

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

Prepared by: **Gustavo Grieco, Jaime Iglesias, Justin Jacob, and Tarun Bansal**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits 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/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits 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 projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits 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.

Trail of Bits 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.

Table of Contents

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of various governance action contracts and the Nitro contracts release 1.0.3-beta.0.

A team of four consultants conducted the review from August 7 to August 11, 2023, for a total of two engineer-weeks of effort. Our testing efforts focused on a number of miscellaneous upgrades to be executed, including smart contract governance and ArbOS updates. With full access to source code and documentation, we performed static testing of the target codebase, using automated and manual processes.

Observations and Impact

The security review discovered only low-severity and informational issues in the upgraded code. We must note, however, that this review covered only governance actions and the changes between specific versions of the Nitro contract, as noted in the coverage section.

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
Low	1
Informational	2

CATEGORY BREAKDOWN

Category	*Count*
Access Controls	2
Undefined Behavior	1

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Mary O'Brien, Project Manager
mary.obrien@trailofbits.com

The following engineers were associated with this project:

Gustavo Grieco, Consultant
gustavo.grieco@trailofbits.com

Jaime Iglesias, Consultant
jaime.iglesias@trailofbits.com

Justin Jacob, Consultant
justin.jacob@trailofbits.com

Tarun Bansal, Consultant
tarun.bansal@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
August 7, 2023	Pre-project kickoff call
August 11, 2023	Delivery of report draft
August 11, 2023	Report readout meeting
August 21, 2023	Delivery of summary report

Project Goals

The engagement was scoped to provide a security assessment of the Offchain Labs governance actions contracts and Nitro contracts. Specifically, we sought to answer the following non-exhaustive list of questions:

- Do all action contracts follow the action contracts standards and guidelines?
- Do all action contracts perform their expected actions?
- Are there any unintentional side effects resulting from the execution of any of the action contracts?
- Are there any unintentional side effects resulting from the changes made to the security council contracts?
- Does the security council activation action perform the expected removal and granting of permissions?
- Has any unexpected behavior been introduced as a result of the changes made to the Nitro contracts?

Project Targets

The engagement involved a review and testing of the targets listed below.

Misc AIP

Repository	https://github.com/OffchainLabs/governance
Version	`29358c27b0f58e26700dc99a9ae9ae8b206cdd51`
Type	Solidity
Platform	Ethereum, Arbitrum

Security Council Elections

Repository	https://github.com/OffchainLabs/governance
Version	`05acd8c80c4232cee70de8784754e51102d2e170`
Type	Solidity
Platform	Arbitrum

ArbOS Upgrade Action Contracts

Repository	https://github.com/OffchainLabs/governance
Version	`cc36c78f95beeb5d4a7de1500e812f2831c31452`
Type	Solidity
Platform	Arbitrum

Nitro Release 1.0.3-beta

Repository	https://github.com/OffchainLabs/nitro-contracts
Version	2ba206505edd15ad1e177392c454e89479959ca5
Type	Solidity
Platform	Ethereum, Arbitrum

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. This review covered the following updates:

- **Miscellaneous action contracts.**
    - **Fix L1 Pricing:** This action contract will update the sequencer's inbox fixed cost associated with including a transaction from 100,000 to 240,000 Ethereum L1 gas units. Additionally, the action contract will disable the "amortization cost cap" of the ArbOS pricing system, which is currently enabled and set to its maximum value; however, to fully disable the feature, the value has to be set to 0.
    - **Set New Sweep Receiver**: This upgrade sets a new receiver address for the unclaimed airdrop tokens to prevent these tokens from being included in the quorum calculation for governance.
    - **Fix Core L1 Timelock Schedule Batch bug**: This upgrade updates the code affected by the incorrect usage of `msg.value` in the `L1ArbitrumTimelock._execute` function.

We reviewed the aforementioned action contracts, first by checking whether they follow the action contract guidelines and then by assessing whether they perform the intended actions and whether any unintended side effects are produced as a consequence of their execution. We also reviewed the updated documentation to be used during and after the system upgrade.

- **Security council election smart contract changes**

  We reviewed a number of small changes to the security council smart contracts. We looked for common flaws that may have been unknowingly introduced, or unexpected changes in the behavior of the contract.

- **Security council activation action contracts**

  Aside from the changes introduced to the security council contracts, we also reviewed a governance action contract that activates the security council elections. This action contract replaces the current 9-of-12 security council with the new one and begins the elections.

  We assessed whether the action contract follows the action contract guidelines and whether the expected permissions are revoked and granted.

- **ArbOS upgrade action contracts**

  This action contract is tasked with upgrading the ArbOS module, first by setting a new WASM root (which identifies the ArbOS version) and then by scheduling the upgrade of the module.

  We assessed whether the action contract follows the action contract guidelines and whether the expected changes were made. Additionally, we studied the upgrade process to check whether any unintended side effects could be produced.

- **Nitro smart contract changes between version v1.0.2 and v1.0.3-beta.0**

  We reviewed a series of small changes and fixes made to the Nitro smart contracts. We looked for common Solidity flaws that may have been introduced and for the introduction of unintended changes in behavior.

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. The following list outlines the coverage limitations of the engagement and indicates system elements that may warrant further review:

- While we have reviewed the code for all upgrade action contracts within the scope, some of the critical values used for them (i.e., contract addresses and configuration parameters) are set upon deployment. Because no deployment scripts were provided, we cannot verify the correctness of these values.

- We have not audited the previous versions of the code, just the changes provided. For instance, in the case of the new release of the Nitro contract, we audited only the changes from v1.0.2 to v1.0.3-beta.0, but did not comprehensively audit any of these versions.

- The ArbOS action review did not include any review of the actual code that will be upgraded, simply the tasks performed by the action contract.

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Vote by signature transactions can fail because of the replay protection	Access Control	**Low**
2	The sweep address is a contract that cannot transfer ETH out	Access Control	**Informational**
3	Outdated governance architecture documentation	Undefined Behavior	**Informational**

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Cross-chain message out-of-order execution could affect sequential proposal execution

Out-of-order execution of outbox transactions on L1 and retryable tickets on L2 can lead to unexpected results when governance proposals rely on the specific ordering of execution of actions.

Since the current governance actions are expected to be submitted in a batch, we include this appendix to describe the risk of using retryables that can be executed out of order. Although this will not affect the current configuration, it can affect future upgrades and should be carefully considered.

Part of the governance system lives in Ethereum mainnet and can involve the use of retryable tickets:

```
/// @dev If the target is reserved "magic" retryable ticket address
address(bytes20(bytes("retryable ticket magic")))
 /// we create a retryable ticket at provided inbox; otherwise, we execute directly
 function _execute(address target, uint256 value, bytes calldata data)
     internal
     virtual
     override
 {
     if (target == RETRYABLE_TICKET_MAGIC) {
         // if the target is reserved retryable ticket address,
         // we retrieve the inbox from the data object and
         // then we create a retryable ticket,
         (
     …
```

Figure C.1: Code to execute proposal that require retryable tickets in `L1ArbitrumTimelock`

At the same time, passed proposals can be executed with a certain order, either in a batch or using the `predecessor` field:

```
/// @inheritdoc TimelockControllerUpgradeable
/// @dev Adds the restriction that only the counterparty timelock can call this func
function scheduleBatch(
    address[] calldata targets,
    uint256[] calldata values,
    bytes[] calldata payloads,
    bytes32 predecessor,
    bytes32 salt,
    uint256 delay
) public virtual override (TimelockControllerUpgradeable) onlyCounterpartTimelock {
    TimelockControllerUpgradeable.scheduleBatch(
        targets, values, payloads, predecessor, salt, delay
```

```
    );
}

/// @inheritdoc TimelockControllerUpgradeable
/// @dev Adds the restriction that only the counterparty timelock can call this func
function schedule(
    address target,
    uint256 value,
    bytes calldata data,
    bytes32 predecessor,
    bytes32 salt,
    uint256 delay
) public virtual override (TimelockControllerUpgradeable) onlyCounterpartTimelock {
    TimelockControllerUpgradeable.schedule(target, value, data, predecessor, salt,
delay);
}
```

Figure C.2: `scheduleBatch` and `schedule` functions in `L1ArbitrumTimelock.sol`

However, a malicious user can leverage the out-of-order execution of retryable tickets to break the assumptions of one proposal's execution following another proposal's execution.

Exploit Scenario

Governance votes for the execution of two proposals: A and B, where A is expected to be executed before B. The first proposal produces a retryable ticket, but the execution of this ticket fails for some reason and needs to be manually redeemed. The L1Timelock contract registered the execution of A as successful and allows the execution of B, which can cause failed upgrades or some other unforeseen consequences.

Recommendations

Short term, consider the following changes for both L1 and L2 timelock contracts:

1. Override the `schedule` and `scheduleBatch` functions of timelock to not accept the `predecessor` argument.
2. Override the `scheduleBatch` to accept only one action in the array.

Long term, carefully read through the imported library code to understand the design decisions and implement customizations to suit your requirements and limitations.