Offchain Labs Arbitrum Chains Genesis File Generator

Security Assessment (Summary Report)

December 1, 2025

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

Prepared by: **Simone Monica and Jaime Iglesias**

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

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

The following engineering director was associated with this project:

Benjamin Samuels, Engineering Director, Blockchain
benjamin.samuels@trailofbits.com

The following consultants were associated with this project:

Jaime Iglesias, Consultant
jaime.iglesias@trailofbits.com

Simone Monica, Consultant
simone.monica@trailofbits.com

Project Timeline

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

Date	Event
October 21, 2025	Delivery of report draft
October 23, 2025	Completion of fix review
December 1, 2025	Delivery of final summary report

Project Targets

The engagement involved reviewing and testing the following target.

Genesis File Generator

Repository	https://github.com/OffchainLabs/genesis-file-generator
Version	`59f69a966d343f089d106c931f2ce9c7ee1218d6`
Type	Solidity
Platform	EVM

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of its genesis file generator.

The genesis file generator aims to provide a way for Arbitrum chains to start with a predefined state.

A team of two consultants conducted the review from October 14 to October 17, 2025, for a total of eight engineer-days of effort. With full access to source code and documentation, we performed static and dynamic testing of the target, using automated and manual processes.

Observations and Impact

The main goal of the review was to assess the correctness of the genesis file generator script and look for potential improvements. Additionally, we analyzed the contracts being deployed on genesis and looked for potential issues related to Arbitrum-specific behavior.

Overall, we found the implementation to be clear and thoughtful, filled with explanatory comments and references to where the constants used for the pre-deployed contracts (such as bytecode or addresses) were taken from and how to reproduce them.

The only concern to note is that some of the contracts rely on Solidity behavior that Arbitrum chains customize (e.g., behavior of the `block.number` opcode), and it needs to be determined whether the customized behavior or the original behavior should be used.

Recommendations

Based on the findings identified during the security review, Trail of Bits recommends that Offchain Labs take the following steps:

- **Remediate the findings disclosed in this report.** These findings should be addressed through direct fixes or broader refactoring efforts.

Summary of Findings

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

ID	Title	Type	Severity
1	Unclear whether Arbitrum-specific behavior of Solidity opcodes should be used	Data Validation	**Undetermined**

Detailed Findings

1. Unclear whether Arbitrum-specific behavior of Solidity opcodes should be used

Severity: **Undetermined**	Difficulty: **Low**
Type: Data Validation	Finding ID: TOB-ARBGG-1
Target: `src/PredeployConstants.sol`	

Description

Arbitrum chains contain several instances of custom behavior; some of these behaviors are related to Solidity opcodes, such as `block.number` (`NUMBER`). In the case of Arbitrum chains, querying the current block number via `block.number` will actually return the block number of the first non-Arbitrum parent chain (e.g., Ethereum for ArbOne).

Some of the pre-deployed contracts (such as `Multicall3` and `CreateX`) rely on `block.number`; however, when they were implemented, they were meant to rely on the opcode's "vanilla behavior" (i.e., the default EVM behavior) and not Arbitrum's custom behavior; therefore, it is important to determine which of the two behaviors they should rely on to prevent unintended behavior.

Finally, note that this specific issue is described in Arbitrum's documentation.

Recommendations

Short term, consider whether these contracts should use the default EVM behavior of the implemented Solidity opcodes or the customized Arbitrum behavior. For example, for the `block.number` opcode, determine whether the first non-Arbitrum parent chain block number (`block.number`) or the L2 block number (`arbBlockNumber`) should be used, and update the constants accordingly.

Long term, thoroughly document any Arbitrum-specific behaviors to inform future protocol updates.

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. Code Quality Issues

This appendix contains findings that do not have immediate or obvious security implications. However, addressing them may enhance the code's readability and may prevent the introduction of vulnerabilities in the future.

- The following comment should refer to `MultisendCallOnly`, not `MultsendCallOnly`.

```
/// @dev MultsendCallOnly v1.4.1 (canonical)
```

Figure B.1: Code comment in src/PredeployConstants.sol#L91

- The following comment should read `ID`, not `If`.

```
# If of the chain you're going to create
```

Figure B.2: Code comment in .env.example#L5

C. Fix Review Results

When undertaking a fix review, Trail of Bits reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not comprehensive analysis of the system.

On October 23, 2025, Trail of Bits reviewed the fixes and mitigations implemented by the Offchain Labs team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

In summary, all the issues described in this report were resolved. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Severity	Status
1	Unclear whether Arbitrum-specific behavior of Solidity opcodes should be used	Undetermined	Resolved

Detailed Fix Review Results

TOB-ARBGG-1: Unclear whether Arbitrum-specific behavior of Solidity opcodes should be used

Resolved. The client provided the following context for this finding's fix status:

> *About the use of that opcode, we are aware of this behavior and we've decided to keep it as is, for the following reasons:*
>
> > *CreateX only uses block.number (and blockhash), along with other properties, to calculate a pseudo-random salt for create2 and create3, in the case where the user doesn't provide one.*
> >
> > *Multicall3 returns block.number on the backward-compatible functions coming from Multicall2, but doesn't return it for the newly added functions. Arbitrum chains already have a canonical Multicall2 and an "arbified" Multicall2 (where they return arbitrum block numbers instead). So users are free to choose the arbified Multicall2 if they wish to obtain arbitrum block.numbers instead.*
> >
> > *Both contracts, in their canonical versions, live in Arbitrum One (1, 2) and seem to be commonly used, which adds to the assumption that tooling is working well around this behavior.*

Finally, modifying the bytecode of these contracts would result in deploying them in different addresses, which might break the assumption that these contracts live in their canonical addresses.

D. Fix Review Status Categories

The following table describes the statuses used to indicate whether an issue has been sufficiently addressed.

Fix Status	
Status	**Description**
Undetermined	The status of the issue was not determined during this engagement.
Unresolved	The issue persists and has not been resolved.
Partially Resolved	The issue persists but has been partially resolved.
Resolved	The issue has been sufficiently resolved.

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 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 with our latest news and announcements, please follow @trailofbits on X or LinkedIn 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

Trail of Bits performed all activities associated with this project in accordance with a statement of work and an agreed-upon project plan.

Security assessment projects are time-boxed and often rely on information 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 software controls and security properties. 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. A project's time and resource constraints also limit their use.