



Nitro Contracts with BoLD

Security Assessment (Summary Report)

October 30, 2024

Prepared for:

Harry Kalodner, Lee Bousfield, Steven Goldfeder, and Ed Felten
Offchain Labs

Prepared by: **Gustavo Grieco, Simone Monica, and Jaime Iglesias**

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.

497 Carroll St., Space 71, Seventh Floor
Brooklyn, NY 11215

<https://www.trailofbits.com>

info@trailofbits.com

Notices and Remarks

Copyright and Distribution

© 2024 by Trail of Bits, Inc.

All rights reserved. Trail of Bits hereby asserts its right to be identified as the creator of this report in the United Kingdom.

This report is considered by Trail of Bits to be business confidential information; it is licensed to Offchain Labs under the terms of the project statement of work and intended solely for internal use by Offchain Labs. Material within this report may not be reproduced or distributed in part or in whole without the express written permission of Trail of Bits.

The sole canonical source for Trail of Bits publications, if published, is the [Trail of Bits Publications page](#). Reports accessed through any source other than that page may have been modified and should not be considered authentic.

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

About Trail of Bits	1
Notices and Remarks	2
Table of Contents	3
Project Summary	4
Project Targets	5
Executive Summary	6
Summary of Findings	7
Detailed Findings	8
1. EOAs addresses can be unexpectedly aliased	8
2. EIP-7702 can break assumptions on address aliasing	10
A. Vulnerability Categories	11

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:

Josselin Feist, Engineering Director, Blockchain
josselin.feist@trailofbits.com

The following consultants were associated with this project:

Gustavo Grieco, Consultant
gustavo.grieco@trailofbits.com

Simone Monica, Consultant
simone.monica@trailofbits.com

Jaime Iglesias, Consultant
jaime.iglesias@trailofbits.com

Project Timeline

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

Date	Event
October 21, 2024	Pre-project kickoff call
October 28, 2024	Delivery of report draft
October 28, 2024	Report readout meeting
October 30, 2024	Delivery of summary report

Project Targets

The engagement involved a review and testing of the following target.

Nitro Contracts

Repository	https://github.com/OffchainLabs/nitro-contracts
Version	acb2fd2703b8bda7c1dc15090d4b09052db4766f
Type	Solidity
Platform	EVM

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of a number of changes to the BoLD contracts.

A team of three consultants conducted the review from October 21, 2024 to October 25, 2024, for a total of 2.6 engineer-weeks of effort. With full access to source code and documentation, we performed a manual review of the code in scope.

Observations and Impact

The scope of the review included only the specific changes made to the BoLD contracts between [6b42a38f](#) (previously audited) and [acb2fd2](#). Some of these changes include EIP-7702 support and migration to anyTrustConfirmer (which we audited separately).

In Ethereum, we classify accounts in two types: externally owned accounts (EOAs) and smart contracts. The main difference between the two is that EOAs cannot have code (or rather, have empty code). EIP-7702 allows EOAs to set their code, which has a number of implications, especially when it comes to the assumptions smart contracts make when different accounts interact with them or checks that Solidity itself implements.

Our testing efforts were focused on identifying possible edge cases related to EIP-7702 support that could lead to unexpected behavior. Some of the areas we explored included address aliasing, retryable tickets, and other general assumptions made by the existing contracts, such as the use of `fromOrigin` methods in the Nitro contracts. We also reviewed the non-EIP-7702-related changes to ensure that no unwanted behavior was introduced.

The review revealed two informational issues related to address aliasing when EIP-7702 is active.

Recommendations

We recommend that the client address the findings presented in this report.

Summary of Findings

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

ID	Title	Type	Severity
1	EOAs addresses can be unexpectedly aliased	Undefined Behavior	Informational
2	EIP-7022 can break assumptions on address aliasing	Data Validation	Informational

Detailed Findings

1. EOAs addresses can be unexpectedly aliased

Severity: Informational

Difficulty: Low

Type: Undefined Behavior

Finding ID: TOB-ARB-1

Target: src/bridge/Inbox

Description

The usage of EIP-7702 when EOAs interact with the Arbitrum contracts in the parent chain can trigger unexpected aliasing.

EIP-7702 allows EOAs to set their code. If an EOA has code, the Arbitrum smart contracts in the parent chain will treat the address as a smart contract. In particular, for deposits, this means that the `depositEth` function will alias its origin address and use it as the destination of the deposit on the L2 side (figure 1.1).

```
function depositEth() public payable whenNotPaused onlyAllowed returns (uint256) {
    address dest = msg.sender;

    // solhint-disable-next-line avoid-tx-origin
    if (AddressUpgradeable.isContract(msg.sender) || tx.origin != msg.sender) {
        // isContract check fails if this function is called during a contract's
        constructor.
        dest = AddressAliasHelper.applyL1ToL2Alias(msg.sender);
    }

    return _deliverMessage(
        L1MessageType_ethDeposit, msg.sender, abi.encodePacked(dest, msg.value),
        msg.value
    );
}
```

Figure 1.1: The `depositEth` function ([src/bridge/Inbox.sol#L202-L214](#))

While this is not a problem, as users can eventually use the `unsafeCreateRetryableTicket` function to move funds, it may appear as unexpected behavior.

Other instances where an EOA address can be potentially aliased are:

- The `msg.sender` of the `depositERC20` function in the `ERC20Inbox` contract
- The `excessFeeRefundAddress` and `callValueRefundAddress` arguments of the `_createRetryableTicket` function in the `AbsInbox` contract

Exploit Scenario

Alice calls `depositEth` from her EOA address with code, expecting that the funds will be deposited to the same address on Arbitrum. However, they are sent to its aliased address.

Recommendations

Short term, consider clearly documenting that behavior in the UI so users are aware of it.

Long term, review the implications of EIP-7702 across all the Arbitrum components.

2. EIP-7702 can break assumptions on address aliasing

Severity: Informational

Difficulty: Low

Type: Data Validation

Finding ID: TOB-ARB-2

Target: src/bridge/Inbox

Description

The usage of EIP-7702 both in the parent and the child chain can result in new types of interactions regarding address aliasing that were not previously possible in practice, and could impact users if they are not aware.

Address aliasing was introduced in the L2 to avoid unexpected or impossible interactions between smart contracts when doing cross-chain transactions.

These assumptions still hold in practice (except if someone manages to find a private key for a smart contract address). However, with the introduction of EIP-7702 in both the parent and child chain, these assumptions are no longer valid in practice.

Specifically, in some cases, EOAs without code are not aliased in the parent chain; however, they could have code in the child chain, which effectively turns them into smart contracts that require address aliasing.

Recommendations

Short term, consider enhancing the documentation around the risks associated with cross-chain calls when users call EOAs with code.

Long term, review the implications of EIP-7702 across all the Arbitrum components.

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.