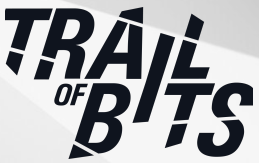

Intuition

Security Assessment (Summary Report)

March 17, 2024

Prepared for:
Billy Luedtke
Intuition

Prepared by: **Michael Colburn and Kurt Willis**

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

Project Summary

Contact Information

The following project manager was associated with this project:

Jeff Braswell, Project Manager
jeff.braswell@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:

Michael Colburn, Consultant
michael.colburn@trailofbits.com

Kurt Willis, Consultant
kurt.willis@trailofbits.com

Project Timeline

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

Date	Event
February 29, 2024	Technical onboarding call
March 7, 2024	Pre-project kickoff call
March 17, 2024	Delivery of report draft
March 17, 2024	Report readout meeting

Project Targets

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

intuition-tob-audit

Repository	https://github.com/0xIntuition/intuition-tob-audit
Version	8e80c751600c481164415f7a7f225217cc164ad2 edc45845db0246c57b538ed5ab6e1f32becba89c
Type	Solidity
Platform	EVM

Executive Summary

Engagement Overview

Intuition engaged Trail of Bits to review the security of its Intuition protocol's smart contracts. The codebase is composed of three core contracts: a standard `TransparentUpgradeableProxy`, a wallet contract `AtomWallet` that leverages ERC-4337 account abstraction, and the main entrypoint and home of most of the protocol logic is the `EthMultiVault` contract, which takes inspiration from the ERC-4626 and ERC-1155 standards.

A team of two consultants conducted the review from March 10 to March 15, 2024, for a total of two engineer-weeks of effort. With full access to source code and documentation, we performed static and dynamic testing of the codebase, using automated and manual processes.

Observations and Impact

As it contains all of the protocol-specific logic, much of our efforts were focused on the `EthMultiVault` contract. We reviewed the bookkeeping, fee calculations, and flow of funds through the vault to identify any inconsistencies in the system's accounting or any known issues pertaining to ERC-4626 vaults. For each of the contracts, we also analyzed them via static analysis with Slither and performed manual review to identify any standard Solidity issues or gaps in the access control mechanisms that could result in an attacker gaining elevated privileges.

Overall, the codebase is quite complex for its size, though by leveraging existing standards for the core building blocks of the contract, this is isolated largely to the protocol-specific logic. We identified five medium-severity, six low-severity, and 10 informational-severity issues as part of this review. Four of the medium-severity issues describe errors in the vault's bookkeeping that result in users receiving too few shares or portions of deposits becoming trapped in the vault. Several of the informational issues identified instances of redundant code or logic that is duplicated across multiple functions that should be consolidated. Due to the number of issues in the bookkeeping and the amount of simplification that could be carried out, additional review may be beneficial.

Recommendations

Take steps to reduce the complexity of the codebase by removing duplicate code, redundant statements and unnecessary features.

Continue to improve the test suite coverage of the protocol to test for realistic scenarios, check for edge cases, explore more advanced testing techniques, and prevent regressions after addressing the issues identified as part of this review.

Consider additional security review after carrying out the recommendations above.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	The contracts only use basic arithmetic operations but apply them in multiple layers that result in fee calculation and bookkeeping logic that is at times difficult to follow. As a result, we identified five issues that would result in an incorrect number of shares being allocated or other bookkeeping errors that result in assets being left unrecoverable in the `EthMultiVault` contract itself.	**Weak**
Auditing	The contracts emit appropriate events for any important state modifying functions. Though compressed atom creation results in the compressed URI being emitted instead of the decoded value, which may be a more useful value.	**Satisfactory**
Authentication / Access Controls	The contracts have a straightforward access control schema with one privileged role in each. We did not identify any issues related to access controls.	**Satisfactory**
Complexity Management	The wallet and proxy contracts are very straightforward. The vault contract contains most of the logic for the protocol. The vault is broken down into logical functions, however there is a very high amount of code duplication across functions that perform similar tasks that should be consolidated into relevant internal helper functions. The fee logic is also fairly complex, with multiple different fees potentially being applied at different times in a transaction on different portions of a deposit. As well, referring to the atom shares purchased through calls to `depositTriple` as an `atomEquityFeeAmount` when this is not actually intended as a fee makes the flow of funds through the protocol more difficult to follow.	**Moderate**

Cryptography and Key Management	This category was not applicable for this review.	**Not Applicable**
Decentralization	The `EthMultiVault` contract is upgradeable. Almost all of the parameters can be updated immediately post-deployment by the contract admin. The protocol funds could be transferred out at any time through an upgrade if the admin's private key becomes compromised. The protocol's documentation indicates this admin will be set to an EOA until sometime post-deployment when it will transition to a multisig. We would strongly suggest migrating to a multisig, even one entirely controlled by core team members, during the production deployment process.	**Weak**
Documentation	The protocol documentation is thorough and includes definitions of protocol-specific terms as well as detailed descriptions of each of the main operations that can be executed through the vault. The contracts themselves also include NatSpec comments and additional helpful inline comments.	**Satisfactory**
Low-Level Manipulation	There is just one line of assembly in the AtomWallet contract that performs error handling after a low-level `call`. There are many other functions in the vault that use `call` to transfer ether between contracts or as part of the redemption process. We also noted one issue related with the use of `abi.encodePacked` and multiple variable length arguments that would result in hash collisions that would allow an attacker to squat on triple identifiers.	**Satisfactory**
Testing and Verification	The codebase has decent test coverage as well as the beginnings of some exploration of invariant testing. However, several of the issues we identified could have been caught by a more comprehensive test suite.	**Moderate**
Transaction Ordering	We did not identify any issues related to transaction ordering. The vault contract uses ghost shares and internal bookkeeping to mitigate the known ERC-4626 share inflation attack.	**Satisfactory**

Summary of Findings

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

ID	Title	Severity
1	createAtomCompressed allows creating duplicate atoms with the same URI	Low
2	Upgrade could lead to mismatch in Atom wallet address prediction	Low
3	Salt contains superfluous address(this)	Informational
4	Unbound storage reads in getVaultStates	Informational
5	EthMultiVault is missing ERC4626 functionality	Informational
6	Protocol deposit fees are unaccounted for in createAtom	Medium
7	createAtom mints sharesForZeroAddress twice	Low
8	Redundant and ineffective reinitialization check	Informational
9	Impossible condition	Informational
10	Triple identifiers can contain hash collisions	Medium
11	EthMultiVault should not receive Ether donations	Low
12	Atom equity should be calculated on raw asset amounts	Medium
13	Distributing atom equity should not include protocol fees	Medium
14	Distributing atom equity should not mint new shares to receiver	Informational
15	Atom wallets can be created before the Atom is created	Low
16	Atom URI data is unbounded	Low

17	getVaultStates does not retrieve counter vaults	**Informational**
18	Excessive duplicate code	**Informational**
19	Admin can bypass fee setter limits	**Informational**
20	Asset accounting should not be reduced by minShare	**Medium**
21	Minting ghost shares is unnecessary to prevent share inflation attacks	**Informational**

A. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories	
Category	**Description**
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication / Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage
Transaction Ordering	The system's resistance to transaction-ordering attacks

Rating Criteria	
Rating	**Description**
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.
Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category is not applicable to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

B. 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.