



Offchain Labs ArbOS 40 Nitro

Security Assessment (Summary Report)

April 28, 2025

Prepared for:

Harry Kalodner, Steven Goldfeder, and Ed Felten

Offchain Labs

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

Table of Contents

Table of Contents	1
Project Summary	2
Project Targets	3
Executive Summary	4
Summary of Findings	5
Detailed Findings	6
1. Arbitrum precompiles break EIP-7702 contract delegation behavior due to non-empty code	6
2. Missing deployment of EIP-2935 contract required for Pectra hard fork compatibility	8
3. Misleading comment in StylusParams storage handler could lead to data corruption	9
4. EIP-2935 block hash contract implementation can be optimized	11
5. Potential error handling issue in Code retrieval after interface change	13
6. EIP-2935 block hash history update function is not called, breaking historical block hash access	14
A. Vulnerability Categories	16
B. Code Quality Findings	18
About Trail of Bits	19
Notices and Remarks	20

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:

Jim Miller, Engineering Director, Blockchain and Cryptography
james.miller@trailofbits.com

The following consultants were associated with this project:

Jaime Iglesias, Consultant
jaime.iglesias@trailofbits.com

Simone Monica, Consultant
simone.monica@trailofbits.com

Nicolas Donboly, Consultant
nicolas.donboly@trailofbits.com

Project Timeline

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

Date	Event
April 7, 2025	Pre-project kickoff call
April 22, 2025	Delivery of report draft
April 22, 2025	Report readout meeting
April 28, 2025	Delivery of final summary report

Project Targets

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

Nitro

Repository	https://github.com/OffchainLabs/nitro
Version	df1fe5636bb0ec4ddae6b8a0eddf8e84a91c3491
Type	Golang
Platform	Arbitrum

go-ethereum

Repository	https://github.com/OffchainLabs/go-ethereum
Version	084f63827520569955a905596878d90d42b734a7
Type	Golang
Platform	Arbitrum

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of ArbOS 40, specifically the commit [df1fe56](#) of the Nitro repo and the commit [084f638](#) of ArbOS's go-ethereum fork.

Additionally, we also reviewed Nitro and Arbitrum's geth fork ([PR#444](#), [PR#3129](#), and [PR#3132](#)), as well as [PR#6](#) of sys-asm.

A team of three consultants conducted the review from April 7, 2025, to April 18, 2025, for a total of six engineer-weeks 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 security assessment focused on evaluating ArbOS v40, which mainly includes changes to support the [Pectra Hardfork](#), merging previous go-ethereum versions into Arbitrum's own fork, and other miscellaneous changes.

The review's main focus was on EIP implementations, precompiles, and the state transition function, as well as on looking for code changes that could potentially be non-backwards compatible or lead to unexpected behavior.

We found one high-severity issue related to EIP-2935 block hash updates and five informational findings. Most issues relate to EIP implementation divergences between Arbitrum and Ethereum. We recommend enhancing the documentation to highlight Arbitrum-specific behaviors.

Recommendations

We recommend addressing the high-severity EIP-2935 implementation issue prior to the ArbOS 40 release. For future protocol upgrades, we suggest developing comprehensive EIP compatibility test suites that specifically validate differences between L1 and L2 implementations. Additionally, we recommend enhancing documentation around Arbitrum-specific behaviors.

Summary of Findings

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

ID	Title	Type	Severity
1	Arbitrum precompiles break EIP-7702 contract delegation behavior due to non-empty code	Undefined Behavior	Informational
2	Missing deployment of EIP-2935 contract required for Pectra hard fork compatibility	Configuration	Informational
3	Misleading comment in StylusParams storage handler could lead to data corruption	Documentation	Informational
4	EIP-2935 block hash contract implementation can be optimized	Configuration	Informational
5	Potential error handling issue in Code retrieval after interface change	Error Reporting	Informational
6	EIP-2935 block hash history update function is not called, breaking historical block hash access	Configuration	High

Detailed Findings

1. Arbitrum precompiles break EIP-7702 contract delegation behavior due to non-empty code

Severity: Informational

Difficulty: Low

Type: Undefined Behavior

Finding ID: TOB-ARBOS40-1

Target: go-ethereum/core/vm/evm.go

Description

EIP-7702 specifies that when a contract delegation points to a precompile address, the call should succeed without execution:

“In case a delegation designator points to a precompile address, retrieved code is considered empty and CALL, CALLCODE, STATICCALL, DELEGATECALL instructions targeting this account will execute empty code, i.e. succeed with no execution given enough gas.”

However, Arbitrum’s implementation does not follow this behavior because its precompiles contain 0xFE (INVALID opcode) instead of empty code.

```
for addr, version := range PrecompileMinArbOSVersions {
    if version == nextArbosVersion {
        stateDB.SetCode(addr, []byte{byte(vm.INVALID)})
    }
}
```

Figure 1.1: Snippet of the UpgradeArbosVersion function
([arbos/arbosState/arbosstate.go#L358-L362](#))

When a contract delegates to a precompile, the resolveCode function returns this non-empty code.

```
if target, ok := types.ParseDelegation(code); ok {
    // Note we only follow one level of delegation.
    return evm.StateDB.GetCode(target)
}
```

Figure 1.2: Snippet of the resolveCode function ([core/vm/evm.go#L625-L628](#))

This causes the EVM interpreter to attempt execution of the INVALID opcode rather than treating it as a successful no-op operation, as required by the EIP.

Recommendations

Short term, implement the behavior defined in EIP-7702 when delegating to a precompile, even for Arbitrum precompiles, or accurately document the different behavior.

Long-term, when implementing EIPs, carefully evaluate how Arbitrum's distinct behavior compared to Ethereum may impact these implementations.

2. Missing deployment of EIP-2935 contract required for Pectra hard fork compatibility

Severity: Informational

Difficulty: Low

Type: Configuration

Finding ID: TOB-ARBOS40-2

Target: arbos/*

Description

The contract required by [EIP-2935](#) (the block hash history contract) has not been deployed. This contract is a necessary component for full Pectra compatibility, as it provides access to historical block hashes beyond the standard 256-block window. Without this deployment, Arbitrum will not fully support the historical block hash functionality introduced in the Pectra hard fork.

Recommendations

Short term, deploy the EIP-2935 contract before upgrading to ArbOS 40 to ensure full compatibility with the Pectra hard fork features.

Long term, establish a pre-upgrade checklist that verifies all required contract deployments and configurations for each Ethereum protocol upgrade, including validation of all dependent EIPs to ensure complete compatibility with Ethereum's protocol evolution.

Notes

This issue was addressed by [PR#3129](#).

3. Misleading comment in StylusParams storage handler could lead to data corruption

Severity: Informational

Difficulty: Low

Type: Documentation

Finding ID: TOB-ARBOS40-3

Target: arbos/programs/params.go

Description

The comment “order matters!” in the Params and Save functions does not immediately convey what it is referring to. By simply examining the code that follows the comment in the Params function, one may assume that it refers to the order of the members of the StylusParams struct; however, it actually refers to the order in which the take closure is used.

In this case, the closure reads encoded values from storage. This means that it is of the utmost importance that the values are consistently read in the same order across code updates; otherwise, a different result will be returned.

Because the comment is not immediately clear, a developer could inadvertently change the order in which the storage is read and written to, potentially causing data corruption on writes or incorrect reads.

```
// order matters!  
stylusParams := &StylusParams{  
    backingStorage: sto,  
    arbosVersion:   p.ArbosVersion,  
    Version:        am.BytesToUint16(take(2)),  
    InkPrice:       am.BytesToUint24(take(3)),  
    MaxStackDepth: am.BytesToUint32(take(4)),  
    FreePages:      am.BytesToUint16(take(2)),  
    PageGas:        am.BytesToUint16(take(2)),  
    PageRamp:       initialPageRamp,  
    PageLimit:      am.BytesToUint16(take(2)),  
    MinInitGas:     am.BytesToUint8(take(1)),  
    MinCachedInitGas: am.BytesToUint8(take(1)),  
    InitCostScalar: am.BytesToUint8(take(1)),  
    CachedCostScalar: am.BytesToUint8(take(1)),  
    ExpiryDays:     am.BytesToUint16(take(2)),  
    KeepaliveDays:  am.BytesToUint16(take(2)),  
    BlockCacheSize: am.BytesToUint16(take(2)),  
}
```

*Figure 3.1: Misleading comment in the Params() function in params.go
([audit-arbitrum-nitro/arbos/programs/params.go#84-102](https://github.com/audit-arbitrum-nitro/arbos/programs/params.go#84-102))*

Recommendations

Short term, rewrite the comments in the Params and Save functions to clarify what they are referring to.

Long term, ensure that all critical comments throughout the codebase are explicit enough for anyone to understand them.

4. EIP-2935 block hash contract implementation can be optimized

Severity: Informational

Difficulty: Low

Type: Configuration

Finding ID: TOB-ARBOS40-4

Target: src/execution_hash/main.eas

Description

The [EIP-2935](#) block hash history contract implementation unnecessarily calls the expensive `%arb_block_num` macro twice, increasing gas costs.

This implementation differs from the standard Ethereum version because it uses an `ArbSys` call to retrieve the block number rather than the native `blocknumber` opcode, making each call more expensive as it requires an additional contract call. This change is necessary because the `blocknumber` opcode does not behave the same as in Ethereum, so the contract needs to be adjusted to provide the expected outcome.

```
;; Check if input is requesting a block hash greater than current block number
;; minus 1.
push 0          ;; [0]
calldataload   ;; [input]
push 1          ;; [1, input]
%arb_block_num ;; [number, 1, input]
sub            ;; [number-1, input]
dup2           ;; [input, number-1, input]
gt            ;; [input > number-1, input]
jumpi @throw   ;; [input]

;; Check if the input is requesting a block hash before the earliest available
;; hash currently. Since we've verified that input <= number - 1, we know
;; there will be no overflow during the subtraction of number - input.
push BUFLen    ;; [buflen, input]
dup2           ;; [input, buflen, input]
%arb_block_num ;; [number, input, buflen, input]
sub            ;; [number - input, buflen, input]
gt            ;; [number - input > buflen, input]
jumpi @throw   ;; [input]
```

*Figure 4.1: %arb_block_num macro is used twice
([sys-asm/src/execution_hash/main.eas#88-107](#))*

Recommendations

Short term, optimize the code by storing the result of the first `%arb_block_num` call on the stack using `dup/swap` operations and reusing it for the second occurrence.

If the code is indeed changed, then update the relevant Nitro bytecode for the system contract.

Notes

This issue was addressed by [PR#6](#). Note that the bytecode still needs to be updated in Nitro.

5. Potential error handling issue in Code retrieval after interface change

Severity: Informational

Difficulty: High

Type: Error Reporting

Finding ID: TOB-ARBOS40-5

Target: `arbos/programs/programs.go`

Description

The transition [in this commit](#) from using `ContractCodeWithPrefix` to `Code` in the `SetProgramCached` function introduces a potential error handling issue. The new `Code` function does not return an error for empty code, unlike the previous implementation, which could lead to silent failures when caching programs.

```
// Not passing in an address is supported pre-Verkle, as in Blockchain's
ContractCodeWithPrefix method.
code, err := db.Reader().Code(common.Address{}, codeHash)
if err != nil {
    return err
}
```

*Figure 5.1: The new Code function does not return an error for empty code
([nitro/arbos/programs/programs.go#452-455](#))*

The new interface was introduced by go-ethereum in [PR#30816](#), which highlights the code semantics change:

“Notably, this interface modifies the function’s semantics. If the contract code is not found, no error will be returned. An error should only be returned in the event of an unexpected issue, primarily for future implementations.”

Recommendations

Short term, add a check for empty code after calling `db.Reader().Code` and return an error if no code is found.

Long term, carefully review go-ethereum function semantic changes for potential impacts on Arbitrum’s implementation when integrating go-ethereum changes into the Arbitrum fork.

6. EIP-2935 block hash history update function is not called, breaking historical block hash access

Severity: High

Difficulty: Low

Type: Configuration

Finding ID: TOB-ARBOS40-6

Target: `arbos/state_processor.go`

Description

The Arbitrum implementation fails to call the `ProcessParentBlockHash` function required by EIP-2935 (Block Hash History).

The function is defined in `go-ethereum` but never called in ArbOS. This is because in `geth`, it is called in the `Process` function that processes new blocks (figure 6.1), while ArbOS uses its own `ProduceBlock` function. This prevents the block hash history contract from being updated with each new block. All calls to retrieve historical block hashes will fail after the `Pectra` hard fork.

```
if p.config.IsPrague(block.Number(), block.Time(), context.ArbOSVersion) ||
p.config.IsVerkle(block.Number(), block.Time()) {
    ProcessParentBlockHash(block.ParentHash(), evm)
}
```

Figure 6.1: `ProcessParentBlockHash` is called by the `Process` function in Arbitrum's `Geth` fork, but is not in ArbOS ([go-ethereum/core/state_processor.go#88-90](https://github.com/offchainlabs/nitro/blob/master/core/state_processor.go#L88-90))

Exploit Scenario

A cross-chain bridge contract relies on historical block hash validation beyond the standard 256-block window. After the `Pectra` hard fork, the bridge tries to access an older block hash using the EIP-2935 mechanism, but since the history contract has not been updated, the call returns incorrect data.

Recommendations

Short term, modify the `ApplyInternalTxUpdate` function to call `ProcessParentBlockHash` when handling the `InternalTxStartBlockMethodID` case to ensure that the history contract is updated with each new block.

Long term, enhance integration testing when merging `geth` upstream changes to verify that the system works as expected, and expand tests to validate new behavior, such as validating that the EIP-2935 functionality works as intended.

Notes

The Offchain Labs team also identified the same issue during their internal review, which was conducted simultaneously with ours.

This issue was addressed by Nitro and Arbitrum's geth fork [PR#444](#), [PR#3129](#), and [PR#3132](#).

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 Findings

The following finding is not associated with any specific vulnerabilities. However, fixing it will enhance code readability and may prevent the introduction of vulnerabilities in the future.

- The `maxWasmSize` parameter (type `uint32`) is cast to `int` in `getWasmFromContractCode` in the following location. This issue could cause an integer overflow on 32-bit systems where `int` is 32 bits. Values greater than 2^{31-1} would become negative, potentially causing unexpected behavior when passed to `DecompressWithDictionary`. While Arbitrum Nitro likely runs only on 64-bit systems, this issue could cause problems if the code were ever to run on 32-bit architectures.
 - [arbos/programs/programs.go#L317](#)

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

© 2025 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.

Trail of Bits considers this report public information; it is licensed to Offchain Labs under the terms of the project statement of work and has been made public at Offchain Labs's request. Material within this report may not be reproduced or distributed in part or in whole without Trail of Bits' express written permission.

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