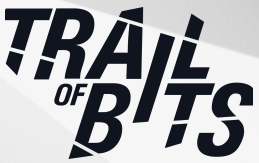

Offchain Labs Arbitrum Mint/Burn Precompile

Security Assessment (Summary Report)

June 2, 2025

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

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

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

Nicolas Donboly, Consultant
nicolas.donboly@trailofbits.com

Project Timeline

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

Date	Event
May 12, 2025	Pre-project kickoff call
May 19, 2025	Delivery of report draft
May 19, 2025	Report readout meeting
May 30, 2025	Completion of fix review
June 2, 2025	Delivery of final summary report

Project Targets

nitro

Repository	https://github.com/OffchainLabs/nitro
Version	PR #3186
Type	Go
Platform	Arbitrum

nitro-contracts

Repository	https://github.com/OffchainLabs/nitro-contracts
Version	PR #335, PR #339
Type	Solidity
Platform	Arbitrum

go-ethereum

Repository	https://github.com/OffchainLabs/go-ethereum
Version	PR #447
Type	Go
Platform	Arbitrum

blockscout

Repository	https://github.com/OffchainLabs/blockscout
Version	PR #197
Type	Elixir
Platform	Arbitrum

nitro-testnode

Repository	https://github.com/OffchainLabs/nitro-testnode
Version	PR #135
Type	TypeScript
Platform	Arbitrum

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of the mint/burn feature and the `ERC20MigrationOutbox` component. The new mint/burn feature of Nitro allows chain owners to mint and burn the network's native token, enabling them to avoid locking up a significant amount of liquidity in a contract on the child chain for transferring the chain's custom gas token between the parent and child chains. The `ERC20MigrationOutbox` component is designed to facilitate the migration of native tokens from the native Arbitrum bridge to an external bridge on orbit chains.

A team of three consultants conducted the review from May 14 to May 16, 2025, for a total of nine 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 security assessment focused on ensuring that the new mint/burn and `ERC20MigrationOutbox` features are implemented securely and function as intended. Our team conducted a comprehensive review of the Solidity contracts and Go implementation that enable authorized entities to mint and burn native gas tokens on Arbitrum chains and to migrate native tokens from the native bridge into a custom bridge implementation.

Our review identified three issues: one low-severity finding related to improper timestamp validation in `SetNativeTokenEnabledFrom` that could unintentionally disable the native token management features (TOB-MINTBURN-1), and two informational-severity findings concerning retroactive timestamp modifications (TOB-MINTBURN-2) and undefined behavior in the usage of the mint/burn feature (TOB-MINTBURN-3). We also provide several code quality recommendations that, while not related to security vulnerabilities, would improve maintainability and user experience (appendix B).

Recommendations

We recommend addressing the three issues disclosed in this report and implementing the recommendations provided in the Code Quality Recommendations appendix.

Summary of Findings

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

ID	Title	Type	Severity
1	Native token feature disabled if SetNativeTokenEnabledFrom's timestamp is set to 0 or a future time	Data Validation	**Low**
2	SetNativeTokenEnabledFrom can be set to a past timestamp	Data Validation	**Informational**
3	Native token owners can still use the mint/burn feature even if it is disabled	Undefined Behavior	**Informational**

Detailed Findings

1. Native token feature disabled if SetNativeTokenEnabledFrom's timestamp is set to 0 or a future time

Severity: **Low**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-MINTBURN-1
Target: `precompiles/ArbOwner.go`	

Description

The `SetNativeTokenEnabledFrom` function allows callers to set a `timestamp` to enable the native token management functionalities. The feature can be explicitly disabled by setting `timestamp` to `0`. Alternatively, if the feature is currently active (meaning its stored enablement time is less than or equal to the current `timestamp`), setting `timestamp` to a future value will also make the feature inactive until that future `timestamp` is reached. This creates an unintended mechanism to disable the native token functionality.

```
if (stored == 0 && timestamp < now+NativeTokenEnableDelay) ||
       (stored > now+NativeTokenEnableDelay && timestamp <
now+NativeTokenEnableDelay) {
       return ErrNativeTokenDelay
}
// If the feature is scheduled to be enabled earlier than the minumum delay,
// then the new time to enable it must be only further in the future.
if stored > now && stored <= now+NativeTokenEnableDelay && timestamp < stored {
       return ErrNativeTokenBackward
}
return c.State.SetNativeTokenEnabledFromTime(timestamp)
```

Figure 1.1: Snippet of the `SetNativeTokenEnabledFrom` function (`precompiles/ArbOwner.go#L81–L90`)

Exploit Scenario

The stored timestamp is `100` and the current block time (now) is `120`, so the feature is active. The chain owner calls `SetNativeTokenEnabledFrom` with `timestamp` set to `130` and makes the feature inactive until block time `130` is reached.

Recommendations

Short term, add validation to prevent callers of `SetNativeTokenEnabledFrom` from setting the feature's enablement timestamp to a future value. Additionally, consider implementing a separate, explicit function for disabling the native token functionality.

Long term, enhance the test suite for setter functions to ensure that corner cases are covered for all possible input values.

2. SetNativeTokenEnabledFrom can be set to a past timestamp

Severity: **Informational**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-MINTBURN-2
Target: `precompiles/ArbOwner.go`	

Description

The `SetNativeTokenEnabledFrom` function lacks validation for backward `timestamp` settings. This allows for the arbitrary change of the activation `timestamp`, `nativeTokenEnabledTime`, to any point in the past.

```
func (con ArbOwner) SetNativeTokenEnabledFrom(c ctx, evm mech, timestamp uint64)
error {
	if timestamp == 0 {
		return c.State.SetNativeTokenEnabledFromTime(0)
	}
	stored, err := c.State.NativeTokenEnabledFromTime()
	if err != nil {
		return err
	}
	now := evm.Context.Time
	// If the feature is disabled, then the time must be at least 7 days in the
	// future.
	// If the feature is scheduled to be enabled more than 7 days in the future,
	// and the new time is also in the future, then it must be at least 7 days
	// in the future.
	if (stored == 0 && timestamp < now+NativeTokenEnableDelay) ||
		(stored > now+NativeTokenEnableDelay && timestamp <
now+NativeTokenEnableDelay) {
		return ErrNativeTokenDelay
	}
	// If the feature is scheduled to be enabled earlier than the minumum delay,
	// then the new time to enable it must be only further in the future.
	if stored > now && stored <= now+NativeTokenEnableDelay && timestamp < stored
{
		return ErrNativeTokenBackward
	}
	return c.State.SetNativeTokenEnabledFromTime(timestamp)
}
```

Figure 3.1: Snippet of the `SetNativeTokenEnabledFrom` function (`precompiles/ArbOwner.go#L63–L91`)

For example, if the current time is `100` and the stored activation timestamp is `90`, the chain owner could reset the activation timestamp all the way back to `1`. This would allow the owner to retroactively modify when the native token functionality was activated.

Recommendations

Short term, add input validation to prevent backward timestamp adjustments.

Long term, enhance the test suite for setter functions to ensure that corner cases are covered for all possible input values.

3. Native token owners can still use the mint/burn feature even if it is disabled

Severity: **Informational**	Difficulty: **High**
Type: Undefined Behavior	Finding ID: TOB-MINTBURN-3
Target: `precompiles/ArbNativeTokenManager.go`	

Description

Native token owners can use the mint/burn feature regardless of whether it is currently enabled.

```
// Mints some amount of the native gas token for this chain to the given address
func (con ArbNativeTokenManager) MintNativeToken(c ctx, evm mech, amount huge) error
{
	if !con.hasAccess(c) {
		return c.BurnOut()
	}
	if err := c.Burn(mintBurnGasCost); err != nil {
		return err
	}

	evm.StateDB.ExpectBalanceMint(amount)
	evm.StateDB.AddBalance(c.caller, uint256.MustFromBig(amount),
tracing.BalanceIncreaseMintNativeToken)
	return nil
}
```

Figure 3.1: The `MintNativeToken` function in (`precompiles/ArbNativeTokenManager.go#24–L35`)

As shown in figure 3.1, the only check performed during minting is whether the caller has access (i.e., whether they are a native token owner); there is no check for whether the feature is currently enabled. It is important to note that a native token owner can be added only by the chain owner and only under the condition that the feature is enabled at the time they are added.

According to the client, the process for disabling the feature involves removing the `NativeTokenManagement` owners and deactivating the feature. It remains active until both steps have been completed. However, there is no documentation currently available for this behavior, so we recommend creating documentation for it.

Recommendations

Short term, consider including additional checks in the `ArbNativeTokenManager` precompiled contract to ensure the feature can be used only when it is enabled.

Long term, thoroughly document the intended behavior of the feature and the flow for enabling and disabling it.

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 Recommendations

The following recommendations are not associated with any specific vulnerabilities. However, addressing them will enhance code readability and may prevent the introduction of vulnerabilities in the future.

- Correct the three typos in these comments:
  - `arbos/arbosState/arbosstate.go#L216`
  - `precompiles/ArbOwner.go#L85`
- Add a specific migration event to the `migrate` function:
  - `src/bridge/extra/ERC20MigrationOutbox.sol#L30–L39`
- Add a public-facing `view` function for users to know whether the native token functionality is active. There is no easy way for users to find this information in the current implementation.
- Move the `IsNativeTokenOwner` and `GetAllNativeTokenOwners` functions to the `ArbOwnerPublic` contract. They are currently in `ArbOwner`, linked below:
  - `precompiles/ArbOwner.go#L114–L122`
- Add zero-address checks in the `ERC20MigrationOutbox` contract's `constructor` function:
  - `src/bridge/extra/ERC20MigrationOutbox.sol#L23–L27`
- Add `NativeTokenMinted` and `NativeTokenBurned` events in the `ArbNativeTokenManager` contract:
  - `src/precompiles/ArbNativeTokenManager.sol#L13–L29`
- Add NatSpec comments to the `IERC20MigrationOutbox` interface functions and errors:
  - `src/bridge/extra/IERC20MigrationOutbox.sol#L10–L15`
- Add a test case for the `executeCall` branch of the `migrate` function, which is not currently tested in `ERC20MigrationOutbox.t.sol`:
  - `src/bridge/extra/ERC20MigrationOutbox.sol#L35–L37`

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 reviewing specific areas of the source code and system configuration, not a comprehensive analysis of the system.

The Offchain Labs team has stated that the three issues disclosed in this report describe intended system behavior and, therefore, has not implemented fixes for these issues. The team did submit PR #3255, PR #3264, and PR #3260, which we reviewed from May 28 to May 30, 2025. The first pull request introduces a restriction for L2-to-L1 messages that prevents withdrawals when the mint/burn feature is enabled. This adds an additional layer of security to prevent situations in which the native bridge becomes undercollateralized. The two other pull requests implement UX improvements recommended in the Code Quality Recommendations appendix.

For additional information, please see the Detailed Fix Review Results below.

ID	Title	Severity	Status
1	Native token feature disabled if SetNativeTokenEnabledFrom's timestamp is set to 0 or a future time	Low	Unresolved
2	SetNativeTokenEnabledFrom can be set to a past timestamp	Informational	Unresolved
3	Native token owners can still use the mint/burn feature even if it is disabled	Informational	Unresolved

Detailed Fix Review Results

TOB-MINTBURN-1: Native token feature disabled if SetNativeTokenEnabledFrom's timestamp is set to 0 or a future time
Unresolved. The client indicated that this is intended behavior.

TOB-MINTBURN-2: SetNativeTokenEnabledFrom can be set to a past timestamp
Unresolved. The client indicated that this is intended behavior.

TOB-MINTBURN-3: Native token owners can still use the mint/burn feature even if it is disabled
Unresolved. The client indicated that this is intended behavior and that token owners will need to be removed as part of the flow to disable the mint/burn feature.

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 to date 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

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.