Orbit and Governance Upgrade Actions v2.1

Security Assessment

August 29, 2024

Prepared for:
Offchain Labs

Prepared by: **Gustavo Grieco**

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

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:

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

Project Timeline

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

Date	Event
August 9, 2024	Pre-project kickoff call
August 16, 2024	Delivery of report draft
August 29, 2024	Delivery of comprehensive report

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of their Orbit governance actions for various upgrades. The upgrades included in this audit are updating the wasm root hash; upgrading the rollup contracts to version 0.2.1; increasing the time delay for governance actions; enabling the fast confirm committee; and migrating any previous AnyTrust fast confirmer.

A team of one consultant conducted the review from August 12, 2024 to August 16, 2024, for a total of one engineer-week of effort. Our testing efforts focused on the manual review of the code that performs the upgrade actions.

Observations and Impact

The identified issues relate to the incorrect usage of a Gnosis Safe multisig. In particular, TOB-ORBUPG-001 allows external users to block an upgrade, requiring redeployment of the action contract to complete the upgrade.

Recommendations

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

- **Carefully monitor the blockchain during the deployment and execution of upgrade actions.** Although the use of an additional salt value on the deployment mitigates TOB-ORBUPG-001, front-running the transaction is possible in certain cases.

- **Properly document how chain owners should use Gnosis Safe multisig for implementing a fast confirmation committee.** This will avoid future issues during deployment and usage of this third-party component.

Finding Severities and Categories

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
High	0
Medium	1
Low	0
Informational	1

CATEGORY BREAKDOWN

Category	*Count*
Configuration	1
Timing	1

Project Goals

The engagement was scoped to provide a security assessment of the Orbit and Governance Upgrade Actions v2.1. Specifically, we sought to answer the following non-exhaustive list of questions:

- Is there any way to block or delay the execution of an upgrade?
- Do the upgrades introduce any security risks?
- Does a storage change of the upgraded version of the rollup code create any potential issues?
- Is the Gnosis Safe multisig correctly configured and used?

Project Targets

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

Orbit Actions

Repository	https://github.com/OffchainLabs/orbit-actions/pull/16 https://github.com/OffchainLabs/orbit-actions/pull/19 https://github.com/OffchainLabs/orbit-actions/pull/20

Version	`b743a21cb1aac7efe50da04dcfa0271c3c3538c8`
	`cc4fc585b1832896d0ed79e457b4dc6003653abc`
	`7157bc16c6505ee3e468bbbe4436df1073c96dee`

Type	Solidity
Platform	EVM

Governance Actions

Repository	https://github.com/ArbitrumFoundation/governance/pull/305 https://github.com/ArbitrumFoundation/governance/pull/306
Version	`e7ffee080a21a0f66b0992f33f3ab885c6b667f3` `f616c311cc2294de6ba6c4b44fc3b2029a672e93`
Type	Solidity
Platform	EVM

Nitro Contracts

Repository	https://github.com/OffchainLabs/nitro-contracts/pull/233
Version	`22d2f322d827b588659486b4b1cf7f81d771350d`
Type	Solidity
Platform	EVM

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following:

- Upgrade Orbit contracts and permissionless enable fast confirmation: These provide a number of state changes in the rollup:
    - Allow Orbit chains to upgrade to v2.1.0 from supported versions. Only relevant contracts (i.e., `ChallengeManager`, `OSP`, `RollupLogics`) are upgraded.
    - Configure `condOsp` to support ongoing challenges using the old `OSP`.
    - Additionally, the `EnableFastConfirmAction` contract provides an easy approach to enable fast confirmation by bundling the setup of all dependencies.
    - Schedule ArbOS 31 Bianca upgrade using the ArbOS upgrade at timestamp action.
    - Enable WASM cache manager using the `AddWasmCacheManagerAction` upgrade action.
- Upgrade Orbit contracts and permissioned enable fast confirmation using a specific address: This action is similar to the previous one, but instead of deploying a Gnosis Safe multisig, it enables fast confirmation using a specific address.
- Updates governance timelock delay: this action introduces an eight-day delay on the execution of governance action, which allows users to withdraw funds if they do not agree with the governance action (or even if it is malicious).
- Migrate `anyTrustFastConfirmer`, if any exists: This action ensures that any previous AnyTrust fast confirmer is migrated.

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. The following list outlines the coverage limitations of the engagement and indicates system elements that may warrant further review:

- We did not review Gnosis Safe multisig code, except for the specific interactions during deployment of the upgrade action.
- We did not review code changes between compatible upgrade versions (e.g. 1.1.0 to 2.1.0), except for their effect on the state compatibility after the upgrade.

Summary of Findings

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

ID	Title	Type	Severity
1	Gnosis safe deployment allows users to disrupt governance action execution	Timing	**Medium**
2	Threshold of signatures used in fastConfirmer can be too inflexible	Configuration	**Informational**

Detailed Findings

1. Gnosis safe deployment allows users to disrupt governance action execution	
Severity: **Medium**	Difficulty: **Low**
Type: Timing	Finding ID: TOB-ORBUPG-001
Target: `contracts/parent-chain/fast-confirm/EnableFastConfirmAction.sol`	

Description

Any user can accidentally or intentionally block the usage of a Gnosis safe deployment to set up a fast confirmer committee as part of the governance action.

The fast confirmer configuration action deploys a Gnosis Safe multisig in order to implement a committee of validators to fast-confirm a rollup state:

```
    function perform(IRollupAdmin rollup, address[] calldata fastConfirmCommittee)
external {
        …
        address fastConfirmer =
IGnosisSafeProxyFactory(GNOSIS_SAFE_PROXY_FACTORY).createProxyWithNonce(
            GNOSIS_SAFE_1_3_0,
            abi.encodeWithSignature(

"setup(address[],uint256,address,bytes,address,address,uint256,address)",
                fastConfirmCommittee,
                fastConfirmCommittee.length,
                address(0),
                "",
                GNOSIS_COMPATIBILITY_FALLBACK_HANDLER,
                address(0),
                0,
                address(0)
            ),
            uint256(keccak256(abi.encodePacked(rollup)))
        );
        rollup.setAnyTrustFastConfirmer(fastConfirmer);
        address[] memory validators = new address[](1);
        validators[0] = fastConfirmer;
        bool[] memory val = new bool[](1);
        val[0] = true;
        rollup.setValidator(validators, val);
```

```
        rollup.setMinimumAssertionPeriod(1);
    }
```

Figure 1.1: Part of the perform function from the Enable Fast Confirmation action

However, since the deployment is performed in a deterministic way using `create2` from a factory contract, the salt must be unique to avoid collisions. Using the same salt as the one from an already-deployed Gnosis Safe will cause this action to revert.

The updated version of the code includes a salt value that mitigates this issue:

```
    function perform(IRollupAdmin rollup, address[] calldata fastConfirmCommittee,
uint256 salt) external {
        …
        address fastConfirmer =
IGnosisSafeProxyFactory(GNOSIS_SAFE_PROXY_FACTORY).createProxyWithNonce(
            GNOSIS_SAFE_1_3_0,
            abi.encodeWithSignature(

"setup(address[],uint256,address,bytes,address,address,uint256,address)",
                fastConfirmCommittee,
                fastConfirmCommittee.length,
                address(0),
                "",
                GNOSIS_COMPATIBILITY_FALLBACK_HANDLER,
                address(0),
                0,
                address(0)
            ),
            salt
        );
```

Figure 1.2: Part of the `perform` function from the fast confirmation action

However, we stress that this is only a mitigation, as front-running this transaction in certain chains like Ethereum mainnet is still feasible.

Additionally, in the `UpgradeAndEnableFastConfirmAction` action, the rollup owner must perform the deployment of the `AnyTrustFast` confirmer:

```
    function perform() external {
        …

        // Setup AnyTrustFastConfirmer
        require(
            IRollupAdminFC(rollupAddress).anyTrustFastConfirmer() == address(0),
            "UpgradeAndEnableFastConfirmAction: Fast confirm already enabled"
        );
        IRollupAdminFC(rollupAddress).setAnyTrustFastConfirmer(
            anyTrustFastConfirmer
```

```
        );
        require(
            IRollupAdminFC(rollupAddress).anyTrustFastConfirmer()
                == anyTrustFastConfirmer,
            "UpgradeAndEnableFastConfirmAction: Unexpected anyTrustFastConfirmer"
        );

        // Set AnyTrustFastConfirmer as validator
        address[] memory validators = new address[](1);
        validators[0] = anyTrustFastConfirmer;
        bool[] memory values = new bool[](1);
        values[0] = true;
        IRollupAdmin(rollupAddress).setValidator(validators, values);
        require(
            IRollupCore(rollupAddress).isValidator(anyTrustFastConfirmer),
            "UpgradeAndEnableFastConfirmAction: Failed to set validator"
        );

        // Set minimum assertion period
        IRollupAdmin(rollupAddress).setMinimumAssertionPeriod(
            newMinimumAssertionPeriod
        );
        require(
            IRollupCore(rollupAddress).minimumAssertionPeriod()
                == newMinimumAssertionPeriod,
            "UpgradeAndEnableFastConfirmAction: Failed to set minimum assertion
period"
        );
    }
```

Figure 1.3: Part of the `perform` function from the fast confirmation and upgrade action

Again, if the deployment uses Gnosis Safe multisig and an attacker is able to guess and front-run the deployment, the result could be catastrophic for the rollup.

Exploit Scenario

A malicious user front-runs the creation of the Gnosis safe deployment, blocking the governance action until it is created again.

Recommendations

Short term, carefully monitor the blockchain during the deployment and execution of this governance action to detect potential front-running attempts.

Long term, review the security assumptions and requirements for third-party code before using it in governance actions.

2. Threshold of signatures used in fastConfirmer is overly inflexible

Severity: **Informational**	Difficulty: **High**
Type: Configuration	Finding ID: TOB-ORBUPG-002
Target: `contracts/parent-chain/fast-confirm/EnableFastConfirmAction.sol`	

Description

The fast confirmer committee is implemented using a Gnosis Safe that uses the maximum threshold, making it very inflexible if changes are needed.

```
function perform(IRollupAdmin rollup, address[] calldata fastConfirmCommittee,
uint256 salt) external {
        …
        address fastConfirmer =
IGnosisSafeProxyFactory(GNOSIS_SAFE_PROXY_FACTORY).createProxyWithNonce(
            GNOSIS_SAFE_1_3_0,
            abi.encodeWithSignature(

"setup(address[],uint256,address,bytes,address,address,uint256,address)",
                fastConfirmCommittee,
                fastConfirmCommittee.length,
                address(0),
                "",
                GNOSIS_COMPATIBILITY_FALLBACK_HANDLER,
                address(0),
                0,
                address(0)
            ),
            salt
        );
       …
```

Figure 2.1: Part of the `perform` function from the fast confirmer action

However, using the maximum threshold while providing maximum protection to the rollup can be overly inflexible, potentially preventing a required administrative action in the Gnosis Safe itself.

Exploit Scenario

A member of the fast confirm committee has their key leaked, destroyed, or revoked. The remaining members of the committee are unable to change the Gnosis Safe multisig configuration since they do not reach the threshold limit.

Recommendations
Short term, allow chain owners to select the threshold number. This issue was fixed and verified during the last part of the review.

Long term, review the security assumptions and requirements for third-party code before using it in governance actions.

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.