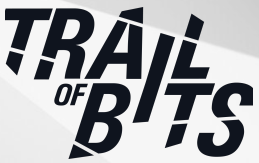

Ritual Infernet SDK Smart Contracts

Security Assessment (Summary Report)

August 29, 2024

Prepared for:
Anish Agnihotri
Ritual

Prepared by: **Justin Jacob and Tarun Bansal**

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

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**
A. Code Quality Findings **8**
B. Fix Review Results **9**
Detailed Fix Review Results 10
C. Fix Review Status Categories **13**

Project Summary

Contact Information

The following project manager was associated with this project:

Sam Greenup, Project Manager
sam.greenup@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:

Justin Jacob, Consultant
justin.jacob@trailofbits.com

Tarun Bansal, Consultant
tarun.bansal@trailofbits.com

Project Timeline

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

Date	Event
June 10, 2024	Pre-project kickoff call
June 17, 2024	Delivery of report draft
June 17, 2024	Report readout meeting
August 29, 2024	Delivery of summary report with fix review

Project Targets

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

Infernet SDK Smart Contracts

Repository	https://github.com/ritual-net/infernet-sdk/
Version	`8e6cd6f5cbd66dc9baacb895a2ed8fe2c9ee3b6f`
Type	Solidity
Platform	Ethereum

Executive Summary

Engagement Overview

Ritual engaged Trail of Bits to review the security of its Infernet SDK smart contracts. The SDK creates an interface for on-chain consumers to specify and pay for data subscriptions from a set of off-chain nodes, which deliver the responses.

A team of two consultants conducted the review from June 11 to June 14, 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 target using automated and manual processes.

Observations and Impact

The Infernet SDK places a high degree of trust in off-chain node operators' honesty and correct validation of consumer trust. Consumers in the system can specify arbitrary subscriptions that could have negative or malicious effects, such as those that maliciously slash nodes and force node operators to waste transaction fees on failed compute delivery. In addition, because node operators and subscription creators are required to escrow their money into the system, there could be financial incentives for misbehavior even if consumers are initially trusted.

Recommendations

Based on the findings identified during the security review, Trail of Bits recommends that Ritual take the following steps to secure the system:

- **Remediate the findings disclosed with this report.** These findings should be addressed as part of direct remediation or as part of any refactoring that may occur when addressing other recommendations.
- **Document all system actors, their privileges, and their incentives.** A strong specification that identifies all system actors, their privileges, and their incentives to behave honestly will be a valuable aid in designing a better incentive mechanism to prevent malicious behavior. The specification can also help with the future development and testing of the system.
- **Document the assumptions made by the off-chain components.** Documentation of the assumptions considered by the off-chain components can highlight potential gaps in the smart contract design and implementations and can highlight edge cases in system behavior.

Summary of Findings

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

ID	Title	Type	Severity
1	The `nodeWallet` argument of the `deliverCompute` function is not validated	Data Validation	**Medium**
2	Consumers can get their work done for free	Data Validation	**Medium**
3	A consumer can use a malicious verifier to slash nodes for valid submissions	Data Validation	**High**
4	Various state-changing operations do not emit events	Auditing and Logging	**Medium**
5	`SafeTransferLib` is not used	Configuration	**Medium**
6	Malicious subscriptions can force nodes to waste gas	Timing	**High**
7	`deliverCompute` can be front-run to steal the payment of node operators	Timing	**High**
8	Anyone can create a wallet for anyone else	Access Control	**Informational**
9	`finalizeProofVerification` can be reentered to drain consumer wallets	Data Validation	**High**

A. Code Quality Findings

This appendix contains findings that do not have immediate or obvious security implications. However, they may facilitate exploit chains targeting other vulnerabilities, become easily exploitable in future releases, or decrease code readability. We recommend fixing the issues reported here.

- **There is an incorrect constant in an inline comment.** The inline comment at line #228 in the `_calculateFee` function indicates an incorrect scaling factor, 1e5. It should be 1e4.
- **The `block.timestamp` value is not cast to `uint32` before the comparison at line #376 of `Coordinator.sol`.** Not casting `block.timestamp` can cause canceled subscriptions to become active after the timestamp crosses the maximum value of the `uint32` type.
- **The documentation and comments of the `requestProofVerification` function do not advise users to implement an access control rule to allow only the `Coordinator` contract to call it.** Without access controls on this function, anyone can execute a DoS attack on verifiers by requesting proof verification with random inputs.

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

On August 21, 2024, Trail of Bits reviewed the fixes and mitigations implemented by the Ritual team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

In summary, of the nine issues described in this report, the Ritual team has resolved four issues, has partially resolved one issue, and has not resolved the remaining four issues. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Status
1	The `nodeWallet` argument of the `deliverCompute` function is not validated	**Resolved**
2	Consumers can get their work done for free	**Unresolved**
3	A consumer can use a malicious verifier to slash nodes for valid submissions	**Unresolved**
4	Various state-changing operations do not emit events	**Resolved**
5	`SafeTransferLib` is not used	**Resolved**
6	Malicious subscriptions can force nodes to waste gas	**Unresolved**
7	`deliverCompute` can be front-run to steal the payment of node operators	**Unresolved**
8	Anyone can create a wallet for anyone else	**Partially Resolved**
9	`finalizeProofVerification` can be reentered to drain consumer wallets	**Resolved**

Detailed Fix Review Results

TOB-RITUAL-1: The `nodeWallet` argument of the `deliverCompute` function is not validated
Resolved in PR #27. The `deliveryCompute` function now validates the `nodeWallet` argument to be a valid `Wallet` created by the `WalletFactory`.

TOB-RITUAL-2: Consumers can get their work done for free
Unresolved. The issue has not been resolved.

The client provided the following context for this finding's fix status:

> *This is a known design decision in the Infernet SDK which has been explicitly documented in the codebase. Unlike capital-inefficient systems that force a minimal `Wallet` balance or commit-reveal based systems that add interactive complexity, we instead relegate to off-chain reputation systems implemented by Infernet Nodes.*
>
> *Infernet Node operators only respond to contracts they trust, simulate transactions, reduce trust in contracts if they suspect frontrunning, and only respond to requestors that have built up a baseline level of trust.*
>
> *Ritual acknowledges the issue but considers this flexibility a part of the intended functionality of the Infernet SDK.*

TOB-RITUAL-3: A consumer can use a malicious verifier to slash nodes for valid submissions
Unresolved. The issue has not been resolved.

The client provided the following context for this finding's fix status:

> *In the Infernet SDK, consumers can express their preferences across a variety of dimensions:*
>
> 1. *How much they would like to pay and when for compute responses*
> 2. *How often they would like to receive compute responses*
> 3. *Which verifier they would like to subject response payout to*
>
> *Similarly, responding Infernet Nodes can express a binary opinion to the consumer preferences:*
>
> 1. *Agreeing to the preferences, computing a response, and submitting an output on-chain*
> 2. *Disagreeing to the preferences and choosing not to submit a response at all*

In this way, Infernet Nodes maintain their opinion over which requests should be responded to. In cases where consumers opt to use an untrusted or malicious verifier, Infernet Nodes simply ignore the request choosing not to submit a response.

The Infernet SDK, unlike most oracle systems, is not one where each node responds to every request. Instead, nodes have full, opt-in choice over which requests to respond to (by default, none). In the future, this trust may be delegated to registries.

Ritual acknowledges the issue but does not consider it to be a vulnerability in the Infernet SDK.

TOB-RITUAL-4: Various state-changing operations do not emit events
Resolved in PR #28. All the state-changing functions now emit events.

TOB-RITUAL-5: `SafeTransferLib` is not used
Resolved in PR #29. The `Wallet` contract now uses the `SafeTransferLib` for ETH and token transfers and balance enquiry.

TOB-RITUAL-6: Malicious subscriptions can force nodes to waste gas
Unresolved. The issue has not been resolved.

The client provided the following context for this finding's fix status:

Responding to compute requests is strictly opt-in in the Infernet system, with Infernet Nodes responding to no requests by default.

In the outlined scenario, a subscription with high frequency and redundancy is created, requesting repetitive compute responses, where a majority of requests fail.

In the system today, Infernet Nodes track failed requests from consumers via their reputation system, optionally choosing to drop a subscription altogether after a number of failed requests. Via this measure, node operators limit gas used towards failed transactions, preventing malicious fund exhaustion attacks.

Ritual acknowledges the issue but does not consider it to be a vulnerability in the Infernet SDK.

TOB-RITUAL-7: `deliverCompute` can be front-run to steal the payment of node operators
Unresolved. The issue has not been resolved.

The client provided the following context for this finding's fix status:

The Infernet SDK does not prevent frontrunning response outputs (or payouts, as identified in TOB-RITUAL-2) as an explicit design decision.

Instead, SDK consumers (and thus, responding nodes making binary decisions over whether to respond) are given full design flexibility. Consumers may:

1. *Choose to prevent frontrunning by electing a trusted set of nodes via the Allowlist*
2. *Include node-specific metadata in response outputs to prevent naive response copying*
3. *Allow frontrunning to benefit from at-least-once delivery to their compute requests*

Ritual acknowledges the issue but does not consider it to be a vulnerability in the Infernet SDK.

TOB-RITUAL-8: Anyone can create a wallet for anyone else
Partially resolved in PR #30. The WalletCreated even now includes the wallet's creator. This allows UI and other integrators to filter wallets based on the creator value to reduce risks. However, it does not fully mitigate the risks of phishing and spoofing attacks.

The client provided the following context for this finding's fix status:

Delegated creation of `Wallet`(s) is useful functionality in the Infernet SDK but we acknowledge Trail of Bit's caution around UI spam and spoofing attack vectors. As such, we have partially resolved the issue in #30 by explicitly logging the address of a `Wallet` creator, recording creation provenance for downstream consumers.

TOB-RITUAL-9: `finalizeProofVerification` can be reentered to drain consumer wallets
Resolved in PR #31. The `finalizeProofVerification` function now follows the check-effects-interactions pattern to prevent reentrancy attacks.

C. 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.