BLOCKAPEX

SMART CONTRACT SECURITY

V 1.0

DATE: 19th AUG 2024

PREPARED FOR: LIGHTLINK HUMMINGBIRD

About BlockApex

Founded in early 2021, is a security-first blockchain consulting firm. We offer services in a wide range of areas including Audits for Smart Contracts, Blockchain Protocols, Tokenomics along with Invariant development (i.e., test-suite) and Decentralized Application Penetration Testing. With a dedicated team of over 40+ experts dispersed globally, BlockApex has contributed to enhancing the security of essential software components utilized by many users worldwide, including vital systems and technologies.

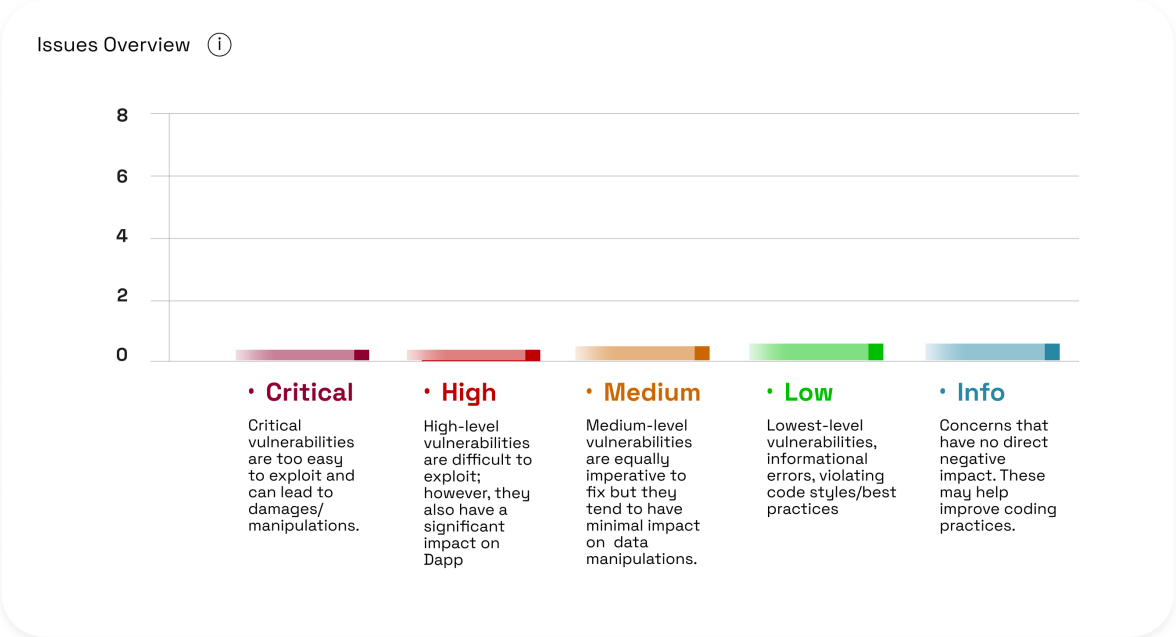
BlockApex has a focus on blockchain security, maintaining an expertise hub to navigate this dynamic field. We actively contribute to security research and openly share our findings with the community. Our work is available for review at our public repository, showcasing audit reports and insights into our innovative practices.

To stay informed about BlockApex's latest developments, breakthroughs, and services, we invite you to follow us on Twitter and explore our GitHub. For direct inquiries, partnership opportunities, or to learn more about how BlockApex can assist your organization in achieving its security objectives, please visit our Contact page at our website , or reach out to us via email at hello@blockapex.io.

Contents

1 Executive Summary

Our team performed a technique called Filtered Audit, where two individuals separately audited the HummingBird Smart Contracts. After a thorough and rigorous manual testing process involving line by line code review for bugs. All the raised flags were reviewed and re-tested to identify any false positives.



1.1 Scope

1.1.1 In Scope

This audit involves a comprehensive review of HummingBird Bridge a fork of the Optimism Bridge, a critical infrastructure component for Layer 2 (L2) scaling solutions on Ethereum. The forked version retains the essential mechanics of Optimism's original design. The core functionality revolves around the secure and efficient transfer of messages and assets (ETH and ERC20 tokens) between Layer 1 (L1) and Layer 2 (L2), ensuring compatibility and security across these layers.

Here's a detailed overview of the business logic for each of the components

L1 Components

1. **L1/L1CrossDomainMessenger.sol**: Facilitates message passing from Layer 1 (L1) to Layer 2 (L2). This contract handles sending and receiving messages between the L1 and L2 messengers, ensuring that messages are relayed with security checks and gas limits to prevent failures or malicious activities. It ensures messages are properly authenticated, relayed, and processed by the L2 counterpart.
2. **L1/L1StandardBridge.sol** : Handles the bridging of assets (ETH and ERC20 tokens) between L1 and L2. It manages the locking and unlocking of tokens on L1 and coordinates with the corresponding L2 bridge to mint or burn tokens as they move between layers. This contract ensures the accurate transfer of value and maintains the integrity of token balances across the bridge.
3. **L1/LightLinkPortal.sol**: The `LightLinkPortal` is a low-level contract responsible for passing messages between Layer 1 (L1) and Layer 2 (L2). It handles the deposit of transactions from L1 to L2 and the finalization of withdrawal transactions from L2 to L1. This contract ensures that messages and transactions are securely relayed between layers, with specific checks to prevent replay attacks and ensure that only valid, proven withdrawals are finalized. The portal also integrates resource metering to monitor and control gas usage, preventing excessive resource consumption during cross-chain operations.
4. **L1/ResourceMetering.sol**: Monitors and enforces resource limits for transactions, such as gas usage. This contract ensures that L1 transactions are executed efficiently and within predefined limits, preventing excessive resource consumption and maintaining network stability.

L2 Components

1. **L2/L1Block.sol**: Stores and manages critical L1 block data on L2. This contract ensures that L2 transactions have access to the most recent L1 block information, enabling accurate cross-layer operations and consistency between L1 and L2.
2. **L2/L2CrossDomainMessenger.sol**: The L2 counterpart to the L1CrossDomainMessenger, this contract handles incoming messages from L1 and facilitates their processing on L2. It ensures that messages from L1 are executed correctly on L2, with appropriate checks and balances to prevent fraudulent or malicious actions.
3. **L2/L2StandardBridge.sol**: Manages the bridging of assets from L2 to L1. It handles the minting and burning of tokens on L2 as assets are transferred across the bridge, ensuring that L2 tokens are correctly accounted for and that the total supply on L2 matches the locked amounts on L1.
4. **L2/L2ToL1MessagePasser.sol**: Facilitates the sending of messages from L2 to L1. This contract stores and processes outgoing messages to be relayed to L1, ensuring that data integrity and message order are maintained when passing information back to L1.

Legacy Components

1. **legacy/AddressManager.sol**: A legacy contract that maps string names to addresses, providing a simple registry for contract addresses. This is used to maintain backward compatibility with older contracts that rely on named address lookups.
2. **legacy/L1ChugSplashProxy.sol**: A legacy proxy contract for L1 that allows for the upgrade of smart contracts by changing their underlying code or storage. This proxy contract includes additional features for managing contract upgrades and ensuring that the system remains upgradable over time.
3. **legacy/ResolvedDelegateProxy.sol**: A proxy contract that uses the AddressManager to resolve implementation addresses dynamically. It is used to maintain backward compatibility with older systems that depend on the AddressManager for managing contract implementations.

Universal Components

1. **Universal/CrossDomainMessenger.sol**: A base contract that provides core functionality for message passing between L1 and L2. It serves as the foundational layer for the L1 and L2 CrossDomainMessenger contracts, providing shared logic for secure and efficient message relay across layers.
2. **universal/ILightLinkMintableERC20.sol**: An interface for the LightLinkMintableERC20 token, defining the standard functions required for minting and burning tokens as part of the cross-chain bridge. This interface ensures that any implementing contract adheres to the required functionality for the bridging process.
3. **universal/LightLinkMintableERC20.sol**: An ERC20 token contract that supports minting and burning operations, specifically designed for cross-chain token transfers. This contract is used to represent assets on L2 that are backed by locked tokens on L1, ensuring a one-to-one peg between layers.
4. **universal/LightLinkMintableERC20Factory.sol** : A factory contract for creating instances of the LightLinkMintableERC20 token. It simplifies the deployment of new ERC20 tokens that are compatible with the bridging system, allowing for streamlined creation and management of bridged assets.
5. **universal/Proxy.sol** : A transparent proxy contract that forwards calls to an implementation contract. It allows for contract upgrades by changing the implementation address without altering the proxy's address, ensuring that the contract can be upgraded without disrupting users.A transparent proxy contract that forwards calls to an implementation contract. It allows for contract upgrades by changing the implementation address without altering the proxy's address, ensuring that the contract can be upgraded without disrupting users.
6. **universal/ProxyAdmin.sol**: Manages the administration of proxy contracts, including upgrading the implementation and managing ownership. It provides a centralized interface for controlling proxy-based upgradeable contracts.
7. **universal/StandardBridge.sol** : The base contract for the L1 and L2 StandardBridge contracts, handling the core logic for bridging assets between layers. It provides shared functionality for managing token transfers, including minting, burning, and locking tokens across L1 and L2.

Contracts in Scope:

- L1/L1CrossDomainMessenger.sol
- L1/L1StandardBridge.sol
- L1/LightLinkPortal.sol
- L1/ResourceMetering.sol
- L2/L1Block.sol
- L2/L2CrossDomainMessenger.sol
- L2/L2StandardBridge.sol
- L2/L2ToL1MessagePasser.sol
- legacy/AddressManager.sol
- legacy/L1ChugSplashProxy.sol
- legacy/ResolvedDelegateProxy.sol
- universal/CrossDomainMessenger.sol
- universal/ILightLinkMintableERC20.sol
- universal/LightLinkMintableERC20.sol
- universal/LightLinkMintableERC20Factory.sol
- universal/Proxy.sol
- universal/ProxyAdmin.sol
- universal/StandardBridge.sol

Initial Commit Hash: 7726b3cad0e375915747a253c5c50c62743d9f6f

1.1.2 Out of Scope

All features or functionalities not delineated within the “In Scope” section of this document shall be deemed outside the purview of this audit.

1.2 Methodology

The codebase was audited using a filtered audit technique. A band of two (2) auditors scanned the codebase in an iterative process for a time spanning 1.5 weeks. Starting with the recon phase, a basic understanding was developed, and the auditors worked on developing presumptions for the developed codebase and the relevant documentation/whitepaper. Furthermore, the audit moved on with the manual code reviews to find logical flaws in the codebase complemented with code optimizations,software, and security design patterns, code styles and best practices

Questions for Security Assessment

1. Is there a robust mechanism to prevent the replay of already processed messages between L1 and L2?
2. How does the system ensure that gas limits are enforced effectively to avoid over-consumption or denial of service?
3. Are ownership and admin privileges appropriately restricted to prevent unauthorized modifications?
4. What safeguards are in place to validate and secure messages passed between L1 and L2?
5. Does the protocol include secure procedures for upgrading proxy contracts to avoid introducing vulnerabilities?
6. How does the protocol ensure withdrawals are only finalized under correct and secure conditions?
7. How does the ResourceMetering component prevent abuse and ensure fair resource allocation?
8. Are there protections against mismanagement or loss of ERC20 tokens during cross-chain transactions?
9. Is there a potential vulnerability within the bridge's architecture or implementation that would allow an attacker to unauthorizedly drain funds from the bridge?
10. Does the bridge possess a sfeguards against the emission of unrestricted deposit events,which could potentially allow for the manipulation or inflation of deposit records without actual asset transfer?
11. Is there an emergency mechanism within the bridge's operational framework that can be activated in response to any potential damages?

1.3 Status Descriptions

Acknowledged: The issue has been recognized and is under review. It indicates that the relevant team is aware of the problem and is actively considering the next steps or solutions.

Fixed: The issue has been addressed and resolved. Necessary actions or corrections have been implemented to eliminate the vulnerability or problem.

Closed: This status signifies that the issue has been thoroughly evaluated and acknowledged by the development team. While no immediate action is being taken.

1.4 Summary of Findings Identified

No Issues Found

2 Findings and Risk Analysis

No Issues Found

Disclaimer:

The smart contracts provided by the client with the purpose of security review have been thoroughly analyzed in compliance with the industrial best practices till date w.r.t. Smart Contract Weakness Classification (SWC) and Cybersecurity Vulnerabilities in smart contract code, the details of which are enclosed in this report.

This report is not an endorsement or indictment of the project or team, and they do not in any way guarantee the security of the particular object in context. This report is not considered, and should not be interpreted as an influence, on the potential economics of the token (if any), its sale, or any other aspect of the project that contributes to the protocol's public marketing.

Crypto assets/ tokens are the results of the emerging blockchain technology in the domain of decentralized finance and they carry with them high levels of technical risk and uncertainty. No report provides any warranty or representation to any third-party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the reports in any way, including to make any decisions to buy or sell any token, product, service, or asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and is not a guarantee as to the absolute security of the project.

Smart contracts are deployed and executed on a blockchain. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. The scope of our review is limited to a review of the programmable code and only the programmable code, we note, as being within the scope of our review within this report. The smart contract programming language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer or any other areas beyond the programming language's compiler scope that could present security risks.

This security review cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While BlockApex has done their best in conducting the analysis and producing this report, it is important to note that one should not rely on this report only - we recommend proceeding with several independent code security reviews and a public bug bounty program to ensure the security of smart contracts