Security Assessment

Chainport Protocol

Jun 18th, 2021

Table of Contents

Summary

This report has been prepared for 2Key to discover issues and vulnerabilities in the source code of the Chainport Protocol project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Chainport Protocol
Description	A cross-chain token bridge implementation
Platform	Ethereum, BSC
Language	Solidity
Codebase	Chainport Smart Contracts
Commit	c0cb0d7b80ee6203131df2a27ca63ed607fbf9be

Audit Summary

Delivery Date	Jul 26, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

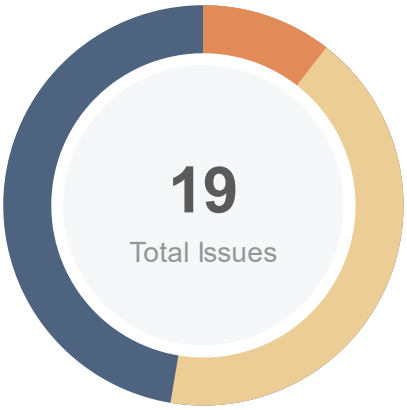
Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	0	0	0	0	0	0
Major	2	0	0	1	0	1
Medium	0	0	0	0	0	0
Minor	8	0	0	4	0	4
Informational	9	0	0	3	6	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum

Findings



Critical	**0** (0.00%)
Major	**2** (10.53%)
Medium	**0** (0.00%)
Minor	**8** (42.11%)
Informational	**9** (47.37%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
BMT-01	Inexistent Zero Address Check	Logical Issue	Minor	Resolved
CAL-01	Volatile Codebase	Coding Style	Minor	Declined
CBB-01	Unutilized Signature Notion	Logical Issue	Minor	Declined
CBE-01	Requisite ERC20 Return Value	Logical Issue	Major	Resolved
CBE-02	Inexistent Input Sanitization	Logical Issue	Minor	Resolved
CBE-03	Unvalidated Nonces	Logical Issue	Minor	Resolved
CBE-04	Incorrect State Transition	Logical Issue	Major	Declined
CBE-05	Unsustainable System	Coding Style	Informational	Acknowledged
CCM-01	Incorrect Minimum Quorum Calculations	Logical Issue	Minor	Declined
CCT-01	Redundant Visibility Specifier	Gas Optimization	Informational	Acknowledged
CTT-01	Centrally Held Total Supply	**Centralization / Privilege**	**Informational**	**Acknowledged**
MRT-01	Variable Visibility Specifier	Coding Style	Informational	Resolved
MRT-02	Inexistent Access Control	Logical Issue	Minor	Declined
MRT-03	Function Visibility Optimization	Gas Optimization	Informational	Acknowledged
MRT-04	Literal Boolean Comparison	Coding Style	Informational	Acknowledged

ID	Title	Category	Severity	Status
MRT-05	System Optimization	Gas Optimization	Informational	Acknowledged
VAL-01	Data Location Optimization	Gas Optimization	Informational	Resolved
VAL-02	Redundant Dynamic Evaluation of Hash	Gas Optimization	Informational	Resolved
VAL-03	Unvalidated Malformed Signature	Logical Issue	Minor	Resolved

BMT-01 | Inexistent Zero Address Check

Category	Severity	Location	Status
Logical Issue	Minor	BridgeMintableToken.sol: 41~48	Resolved

Description

The `setBinanceBridgeContract` function accepts an `address` argument but does not sanitize its validity.

Recommendation

We advise a zero-address check to be imposed on the said argument to ensure no misconfiguration of the contracts can occur.

Alleviation

A zero-address check was introduced by the client alleviating this exhibit.

CAL-01 | Volatile Codebase

Category	Severity	Location	Status
Coding Style	Minor	libraries/Call.sol: 12~179	Declined

Description

The `Call` library is relatively out of line with the rest of the codebase in terms of coding style and performs multiple calculations that could result in different outputs depending on the canonical programming system utilized.

Recommendation

We advise the assembly blocks to be swapped by actual code or that they are made safer, i.e. by not assuming the assignment to `uint8` will properly truncate the remaining bits in the 32-byte word and doing an `AND` operation. Additionally, the signature validation components are undocumented and do not conform to a particular coding style. We strongly recommend the overall implementation to be refactored or downright removed given that it remains unutilized within the codebase.

Alleviation

The client did not acknowledge our concerns and has left the codebase as is.

CBB-01 | Unutilized Signature Notion

Category	Severity	Location	Status
Logical Issue	Minor	ChainportBridgeBsc.sol: 12	Declined

Description

The `signatureValidator` variable remains unutilized in the codebase.

Recommendation

We advise the variable's necessity to be evaluated and potentially omitted if deemed unnecessary.

Alleviation

The client did not perform any alleviation for this exhibit.

CBE-01 | Requisite ERC20 Return Value

Category	Severity	Location	Status
Logical Issue	Major	ChainportBridgeEth.sol: 144, 145, 173, 174, 200, 201, 251, 252, 270, 271	Resolved

Description

Although the ERC20 standard denotes that `transfer` and `transferFrom` invocations should yield a `bool`, not all tokens are compliant (i.e. Tether USDT) and thus would cause the code blocks of the contract to throw.

Recommendation

We strongly recommend a wrapper library like Open Zeppelin's `SafeERC20` to be utilized that opportunistically validates the result of token transfers if it exists.

Alleviation

All ERC-20 transfer calls were replaced by their `safe` prefixed counterparts.

CBE-02 | Inexistent Input Sanitization

Category	Severity	Location	Status
Logical Issue	Minor	ChainportBridgeEth.sol: 121~131	Resolved

Description

The `constructor` of the contract properly sanitizes the `safetyThreshold` but its setter function does not.

Recommendation

We advise the same restrictions to be imposed on the setter function.

Alleviation

A range check was properly introduced for the `safetyThreshold` variable thus alleviating this exhibit.

CBE-03 | Unvalidated Nonces

Category	Severity	Location	Status
Logical Issue	Minor	ChainportBridgeEth.sol: 197, 232	Resolved

Description

The `nonce` value utilized by `releaseTokensTimelockPassed` and `releaseTokens` is not sanitized.

Recommendation

We advise some form of sanitization to occur for the `nonce`, potentially the same one applied in `releaseTokensByMaintainer`, to ensure proper invocations of the function.

Alleviation

The nonce is now properly validated and marked as consumed within the codebase.

CBE-04 | Incorrect State Transition

Category	Severity	Location	Status
Logical Issue	Major	ChainportBridgeEth.sol: 150~177	Declined

Description

The `releaseTokensByMaintainer` does not properly reset the `isTokenHavingPendingWithdrawal` flag or `tokenToPendingWithdrawal` data points.

Recommendation

We advise such a deletion to be introduced to ensure sane state transitions.

Alleviation

The client did not perform any remediation for this exhibit.

CBE-05 | Unsustainable System

Category	Severity	Location	Status
Coding Style	Informational	ChainportBridgeEth.sol: 11~307	Acknowledged

Description

The system is only capable of processing a single withdrawal per token at any given point in time and contains a lot of redundancy within the code.

Recommendation

We strongly recommend the code to be refactored to minimize duplication and enhance its throughput.

Alleviation

The client did not perform any remediations for this exhibit.

CCM-01 | Incorrect Minimum Quorum Calculations

Category	Severity	Location	Status
Logical Issue	Minor	governance/ChainportCongressMembersRegistry.sol: 128~168	Declined

Description

The minimum quorum calculated whenever a member is added or removed overwrites the manually set quorum and sets it to a very high number.

Recommendation

We recommend the rationale around quorum calculations to be expanded upon and if deemed incorrect to be revised as a manually set quorum can cause the calculations to misbehave.

Alleviation

The client did not perform any remediation for the above finding.

CCT-01 | Redundant Visibility Specifier

Category	Severity	Location	Status
Gas Optimization	Informational	governance/ChainportCongress.sol: 15	Acknowledged

Description

The linked variable is redundantly declared as `public`.

Recommendation

We strongly recommend it to be set as `internal` or `private` as it serves no purpose to external parties.

Alleviation

The client did not update the codebase according to our recommendation.

CTT-01 | Centrally Held Total Supply

Category	Severity	Location	Status
Centralization / Privilege	**Informational**	token/ChainportToken.sol: 57	**Acknowledged**

Description

The total supply of the token is minted in full to the `beneficiary` specified during construction.

Recommendation

We advise proper due diligence to be applied as to who this address will be and we recommend it to point to a multi-signature address to prevent single-party malicious actions.

Alleviation

The client has opted not to make any code changes with regards to this finding.

MRT-01 | Variable Visibility Specifier

Category	Severity	Location	Status
Coding Style	Informational	MaintainersRegistry.sol: 12	Resolved

Description

The `_isMaintainer` mapping has no visibility specifier explicitly set.

Recommendation

We advise one to be introduced to ensure consistency in the codebase's outputs. Furthermore, we advise the variable to be renamed to omit the underscore (`_`) character and instead be declared as `public`, thereby rendering the `isMaintainer` getter redundant and safe to remove.

Alleviation

A `private` visibility specifier was introduced to the variable.

MRT-02 | Inexistent Access Control

Category	Severity	Location	Status
Logical Issue	Minor	MaintainersRegistry.sol: 33~46	Declined

Description

The `initialize` function enforces no access control on its first invocation, thereby introducing a race condition.

Recommendation

We strongly recommend some form of access control to be imposed on the function.

Alleviation

The client did not provide any alleviation for this exhibit.

MRT-03 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	MaintainersRegistry.sol: 33~38	Acknowledged

Description

The `initialize` function is solely invoked externally in the codebase.

Recommendation

We strongly recommend it to be set as `external` and its data location arguments to be set as `calldata` greatly optimizing the function's gas cost.

Alleviation

Our recommendation was not applied to the codebase.

MRT-04 | Literal Boolean Comparison

Category	Severity	Location	Status
Coding Style	Informational	MaintainersRegistry.sol: 66, 87	Acknowledged

Description

The linked `require` checks perform a literal comparison of a `bool` variable and a `bool` value (`true`/`false`).

Recommendation

We recommend the equalities to be omitted entirely and the `bool` variables to be used directly either as is or by negating them (`!`).

Alleviation

The client did not perform any adjustments to the codebase for this exhibit.

MRT-05 | System Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	MaintainersRegistry.sol: 61~112	Acknowledged

Description

The system currently retains an array of addresses listed as maintainers and a `bool` mapping indicating whether an address is included in the array or not.

Recommendation

We recommend the mapping to be converted to a `uint256` one directly indicating the index within the array, thereby significantly optimizing removals.

Alleviation

The client chose not to apply our recommendation.

VAL-01 | Data Location Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	Validator.sol: 49	Resolved

Description

The `verifyWithdraw` function is an `external` function with a `memory` argument.

Recommendation

We advise the `memory` argument to be set as `calldata` greatly optimizing its gas cost.

Alleviation

The data location for the argument was safely adjusted to `calldata`.

VAL-02 | Redundant Dynamic Evaluation of Hash

Category	Severity	Location	Status
Gas Optimization	Informational	Validator.sol: 84	Resolved

Description

The linked `keccak256` invocation will always yield the same output.

Recommendation

We advise the result to be stored in a contract-level `constant` thereby optimizing the function's gas cost.

Alleviation

The value is now stored in a contract-level constant variable.

VAL-03 | Unvalidated Malformed Signature

Category	Severity	Location	Status
Logical Issue	Minor	Validator.sol: 153	Resolved

Description

The `ecrecover` invocation's result is directly returned by the function with no form of sanitization being performed.

Recommendation

We advise the return of `ecrecover` to be evaluated against the zero-address as a zero-address return indicates the signature was malformed, a case which does not cause the system to throw. We investigated the other contracts of the system and they always validate a non-zero address, however, this check should still be enforced as best security practice.

Alleviation

The return value of `ecrecover` is now properly validated against the zero address prohibiting invalid signatures.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.