CERTIK

Security Assessment

ButtonWood Protocol

Jul 30th, 2021

Table of Contents

Summary

This report has been prepared for Prometheus Research Labs to discover issues and vulnerabilities in the source code of the ButtonWood 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 major to informational. We recommended addressing these findings to ensure a high level of security standards and industry practices. The team has alleviated all issues in the following iterations: tranche pull_11 and button-wrappers pull 29 pull_30.

Overview

Project Summary

Project Name	ButtonWood Protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/buttonwood-protocol/button-wrappers https://github.com/buttonwood-protocol/tranche
Commit	47dfd232414f63e7b9541c01d76dee1e9028af47, 199f077ac9caee7bbf7939f7ce89c1f12fa241ff

Audit Summary

Delivery Date	Jul 30, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

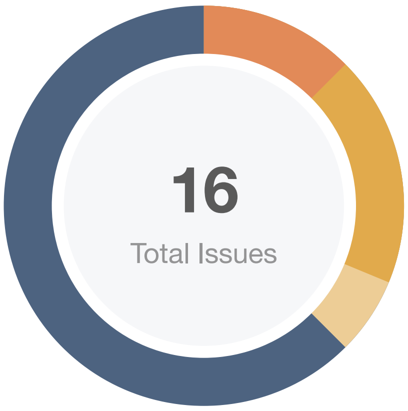
Vulnerability Summary

Vulnerability Level	Total	ⓘ Pending	Partially Resolved	✓ Resolved	ⓘ Acknowledged	⊗ Declined
● Critical	0	0	0	0	0	0
● Major	2	0	0	2	0	0
● Medium	3	0	0	3	0	0
● Minor	1	0	0	0	1	0
● Informational	10	0	0	8	2	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID			File	SHA256 Checksum

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**2** (12.50%)
🟨 **Medium**	**3** (18.75%)
🟨 **Minor**	**1** (6.25%)
🟦 **Informational**	**10** (62.50%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
BCC-01	Missing Zero Amount Check	Logical Issue	● Medium	✓ Resolved
BCC-02	Redundant Variable Initialization	Gas Optimization, Language Specific	● Informational	✓ Resolved
BCC-03	Missing Zero Address Check	Logical Issue	● Informational	✓ Resolved
BCC-04	Ineffiecient Mutex	Logical Issue	● Major	✓ Resolved
BFC-01	Variable Mutability	Gas Optimization	● Informational	✓ Resolved
BTC-01	Redundant Import	Language Specific	● Informational	✓ Resolved
BTC-03	Redundant Require	Language Specific, Logical Issue	● Informational	✓ Resolved
BTC-04	Missing Checks	Logical Issue	● Informational	ⓘ Acknowledged
BTC-05	Bad Function Name	Language Specific, Coding Style	● Informational	✓ Resolved
BTC-06	Missing Approval Event Emission	Language Specific	● Major	✓ Resolved
BTC-07	Local Variable Explicit Return	Coding Style	● Informational	ⓘ Acknowledged
BTC-08	Code Inconsistency	Logical Issue	● Minor	ⓘ Acknowledged
COC-01	Variable Mutability	Language Specific, Gas Optimization	● Informational	✓ Resolved
COC-02	Outdated Interface	Language Specific	● Medium	✓ Resolved

ID	Title	Category	Severity	Status
COC-03	Chainlink Result Not Sanitised Properly	Volatile Code	● Medium	✓ Resolved
TCK-01	Missing Zero Address Check	Logical Issue	● Informational	✓ Resolved

BCC-01 | Missing Zero Amount Check

Category	Severity	Location	Status
Logical Issue	● Medium	tranche-contracts/BondController.sol: 68	✓ Resolved

Description

The linked function does not perform a zero amount check against the argument.

Recommendation

Consider checking the amount against the zero value case.

Alleviation

The team fixed the issue in commit pull 11.

BCC-02 | Redundant Variable Initialization

Category	Severity	Location	Status
Gas Optimization, Language Specific	● Informational	tranche-contracts/BondController.sol: 47, 74, 146	✓ Resolved

Description

The linked variable is redundantly initialised with a value of 0.

Recommendation

Consider removing the redundant initialisation with value 0 as the default value for uint256 is 0.

Alleviation

The team fixed the issue in commit pull 11.

BCC-03 | Missing Zero Address Check

Category	Severity	Location	Status
Logical Issue	● Informational	tranche-contracts/BondController.sol: 37	✓ Resolved

Description

The linked function does not perform a zero address check against the arguments to ensure a proper contract initialization.

Recommendation

Consider checking the variables against the zero address case.

Alleviation

The team fixed the issue in commit pull 11.

BCC-04 | Ineffiecient Mutex

Category	Severity	Location	Status
Logical Issue	● Major	tranche-contracts/BondController.sol: 96	✓ Resolved

Description

The linked function contains an inefficient mutex as solidity functionality can be re-entered.

Recommendation

Consider moving the L122 right after the require check to protect against re-entrancy.

Alleviation

The team fixed the issue in commit pull 11.

BFC-01 | Variable Mutability

Category	Severity	Location	Status
Gas Optimization	● Informational	tranche-contracts/BondFactory.sol: 14, 15	✓ Resolved

Description

The linked variable can be declared immutable.

Recommendation

Consider changing the variables mutability.

Alleviation

The team fixed the issue in commit pull 11.

BTC-01 | Redundant Import

Category	Severity	Location	Status
Language Specific	● Informational	button-contracts-pull7/ButtonToken.sol: 6	✓ Resolved

Description

The linked code imports redundantly the SafeMath library.

Recommendation

Consider removing this import as solidity versions 0.8.x have safe math functionality implemented.

Alleviation

The team fixed the issue in commit pull 29.

BTC-03 | Redundant Require

Category	Severity	Location	Status
Language Specific, Logical Issue	● Informational	button-contracts-pull7/ButtonToken.sol: 151	✓ Resolved

Description

The linked require check is redundant, as the contract is deployed in a controlled environment.

Recommendation

Consider removing the require check or providing a rationale for future use.

Alleviation

The team fixed the issue in commit pull 29.

BTC-04 | Missing Checks

Category	Severity	Location	Status
Logical Issue	● Informational	button-contracts-pull7/ButtonToken.sol: 143~147	ⓘ Acknowledged

Description

The linked constructor needs input sanitisation as address arguments cannot be equal to the zero address plus string arguments cannot be empty.

Recommendation

Consider implementing the checks to ensure proper construction.

Alleviation

The team acknowledged the issue and opted not to alleviate in the current iteration.

BTC-05 | Bad Function Name

Category	Severity	Location	Status
Language Specific, Coding Style	● Informational	button-contracts-pull7/ButtonToken.sol: 551	✓ Resolved

Description

The _activeBits() function ignores the solidity naming conventions (public function beginning with an underscore). Additionally, the function is declared in the private section of the code block as stated in the comments.

Recommendation

Consider renaming the function with respect to solidity best practices and standards.

Alleviation

The team fixed the issue in commit pull 29.

BTC-06 | Missing Approval Event Emission

Category	Severity	Location	Status
Language Specific	● Major	button-contracts-pull7/ButtonToken.sol: 361, 385	✓ Resolved

Description

The linked code does not emit an approval event.

Recommendation

Consider implementing the event emision.

Alleviation

The team fixed the issue in commit pull 29.

BTC-07 | Local Variable Explicit Return

Category	Severity	Location	Status
Coding Style	● Informational	button-contracts-pull7/ButtonToken.sol: 475, 497, 517	ⓘ Acknowledged

Description

The linked statements return the local cAmount variable explicitly.

Recommendation

Consider that the cAmount could be a named variable instead.

Alleviation

The team acknowledged the issue and opted not to alleviate in the current iteration.

BTC-08 | Code Inconsistency

Category	Severity	Location	Status
Logical Issue	● Minor	button-contracts-pull7/ButtonToken.sol: 473	ⓘ Acknowledged

Description

The linked code is inconsistent, as the event is emitted after the external call only here.

Recommendation

Consider refactoring the code according to the pattern.

Alleviation

The team acknowledged the issues and opted not to alleviate in the current iteration.

COC-01 | Variable Mutability

Category	Severity	Location	Status
Language Specific, Gas Optimization	● Informational	button-contracts-pull7/oracles/ChainlinkOracle.sol: 14	✓ Resolved

Description

The linked variable can be declared immutable.

Recommendation

Consider changing the variable mutability.

Alleviation

The team alleviated the issue in pull 30.

COC-02 | Outdated Interface

Category	Severity	Location	Status
Language Specific	● Medium	button-contracts-pull7/oracles/ChainlinkOracle.sol: 5	✓ Resolved

Description

The linked code contains an outdated oracle interface.

Recommendation

Consider implementing the latest oracle interface.

Alleviation

The team alleviated the issue in pull 30.

COC-03 | Chainlink Result Not Sanitised Properly

Category	Severity	Location	Status
Volatile Code	● Medium	button-contracts-pull7/oracles/ChainlinkOracle.sol: 25~28	✓ Resolved

Description

The chainlink oracle price result is not sanitised properly.

Recommendation

Consider that the implementation by Chainlink does not validate against data staleness. In short, one needs to simply check the round deviation between roundID (representing the current round) and answeredInRound (representing the round the price was reported) ensuring that it is within an acceptable threshold.

Alleviation

The team alleviated the issue in pull 30.

TCK-01 | Missing Zero Address Check

Category	Severity	Location	Status
Logical Issue	● Informational	tranche-contracts/Tranche.sol: 32	✓ Resolved

Description

The linked function does not perform a zero address check against the arguments to ensure a proper contract initialization.

Recommendation

Consider checking the variables against the zero address case.

Alleviation

The team fixed the issue in commit pull 11.

Appendix

Finding Categories

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

CERTIK