Blaize.Security

January 18th, 2022 / V. 1.0

CRYPTOBEAR SMART CONTRACT AUDIT

TABLE OF CONTENTS

AUDIT RATING

The CryptoBear smart contract's source code was taken from the files provided by the CryptoBear team

SCORE **9.8**/10

The scope of the project is **CryptoBear** set of contracts**:**

1/ CryptoBearWatchClub

2/ Arkouda

Arkouda.sol primary SHA256:

- A81591B0A9F9AF9733CE5DE5DF178042C83702B86FF80E208A5A2ADAF22179BA

CryptoBearsWatchClub.sol primary SHA256:

- 596533943DB54397E5F625A0A5CF6CFE52F698BB26DF330AF41836C020013C25

Arkouda.sol last SHA256:

- 348A999C4CF24B8A0371E48AE0FC9BF2CFEFD8C426DA43C019F2FA12955ADFC3

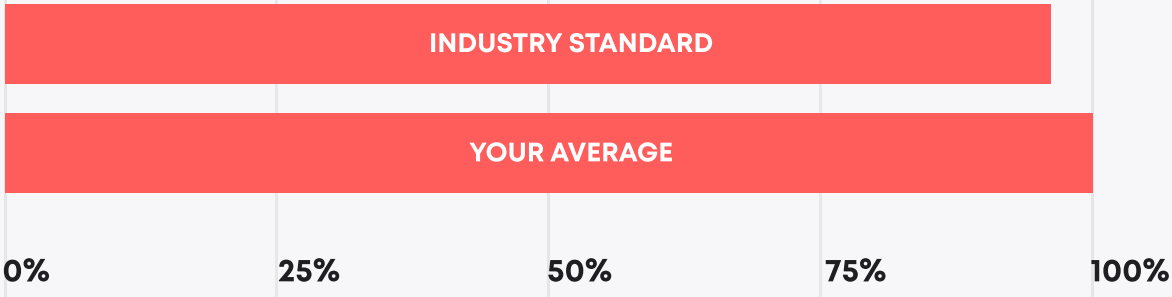
CryptoBearsWatchClub.sol last SHA256:

- 9B6C677F2899DA5F8D670D60AA1694499700142C509E790C658473D64A80C492

TECHNICAL SUMMARY

In this report, we consider the security of the contracts for CryptoBear protocol. Our task is to find and describe security issues in the smart contracts of the platform. This report presents the findings of the security audit of **CryptoBear** smart contracts conducted between **January 13th, 2022 - January 18th, 2022.**

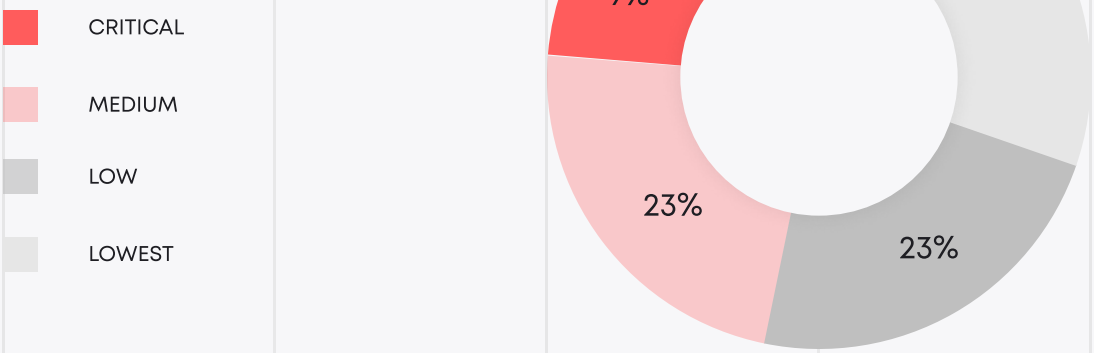
Testable code



The testable code is 98.11%, which is above the industry standard of 95%.

The scope of the audit includes the unit test coverage, that bases on the smart contracts code, documentation and requirements presented by the CryptoBear team. Coverage is calculated based on the set of Truffle framework tests and scripts from additional testing strategies. Though, in order to ensure a security of the contract Blaize.Security team recommends the CryptoBear team put in place a bug bounty program to encourage further and active analysis of the smart contracts.

THE GRAPH OF VULNERABILITIES DISTRIBUTION:



The table below shows the number of found issues and their severity. A total of 9 problems were found. 8 issues were fixed or verified by the CryptoBear team.

	FOUND	FIXED/VERIFIED
Critical	1	1
High	0	0
Medium	2	2
Low	2	2
Lowest	4	3

SEVERITY DEFINITION

Critical

A system contains several issues ranked as very serious and dangerous for users and the secure work of the system. Needs immediate improvements and further checking.

High

A system contains a couple of serious issues, which lead to unreliable work of the system and migh cause a huge information or financial leak. Needs immediate improvements and further checking.

Medium

A system contains issues which may lead to mediumfinancial loss or users' private information leak. Needs immediate improvements and further checking.

Low

A system contains several risks ranked as relatively small with the low impact on the users' information and financial security. Needs improvements.

Lowest

A system does not contain any issue critical to the secure work of the system, yet is relevant for best

AUDITING STRATEGYAND TECHNIQUES APPLIED \ PROCEDURE

We have scanned this smart contract for commonly known and more specific vulnerabilities:

- Unsafe type inference;
- Timestamp Dependence;
- Reentrancy;
- Implicit visibility level;
- Gas Limit and Loops;
- Transaction-Ordering Dependence;
- Unchecked external call - Unchecked math;
- DoS with Block Gas Limit;
- DoS with (unexpected) Throw;
- Byte array vulnerabilities;
- Malicious libraries;
- Style guide violation;
- ERC20 API violation;
- Uninitialized state/storage/ local variables;
- Compile version not fixed.

Procedure

In our report we checked the contract with the following parameters:

- Whether the contract is secure;
- Whether the contract corresponds to the documentation;
- Whether the contract meets best practices in efficient use of gas, code readability;

Automated analysis:

Scanning contract by several public available automated analysis tools such as Mythril, Solhint, Slither and Smartdec. Manual verification of all the issues found with tools.

Manual audit:

Manual analysis of smart contracts for security vulnerabilities. Checking smart contract logic and comparing it with the one described in the documentation.

EXECUTIVE SUMMARY

The contract contained critical issue connected to the incorrect funds flow - it allowed user to claim rewards in spite of contract rules. Though, the team has fixed the issue.

All other issues were connected to missed checks, which may block the contract, and code quality. Nevertheless, all security risk issues were fixed by the team.

The overall security is high enough though the code lacks of readability and the overal quality may be increased. Nevertheless, it performs all desired actions and has solid functionality.

	RATING
Security	9.8
Gas usage and logic optimization	9.8
Code quality	9.5
Test coverage	10
Total	9.8

COMPLETE ANALYSIS

CRITICAL	✔ Resolved

Lack of check

Arkouda::Line 98
There is a check if block.timestamp is greater than rewardGenerationStarTtime but there is no check if rewardGenerationStartime has been set. It can cause premature rewards activation.

Recommendation:

Add a suitable check.

MEDIUM	✔ Resolved

Unheld exception

CryptoBearWatchClub.sol::Line 118, 137
Auction cannot last more than 9300 seconds. After it, the price deduction will become greater than the starting price that will cause a math underflow of unsigned integer and revert.

Recommendation:

Handle this exception if it's planned so, or remake the algorithm.

MEDIUM ✔ Resolved

Unreachable code

Arkouda.sol::Line 147
block.timestamp will never be less than rewardGenerationStartTime so it will never trigger “return 0”.

Recommendation:

Remake the condition

LOW ✔ Resolved

Versioning

Provided contracts use Solidity 0.8.0 which is not the actual version.

Recommendation:

Use stable (without ^) and the actual(0.8.11) version of Solidity.

LOW ✔ Resolved

Meaningless operation

CryptoBearWatchClub.sol::Line 156, 207
There is no need to use additional memory to store msg.sender address. It will cause gas loss.

Recommendation:

Don’t store msg.sender in a variable.

LOWEST	✔ Resolved

Constants can be used

CryptoBearWatchClub.sol::Lines 31, 32
If a variable's state does not change in functionality - it can be declared as constant to save gas on deployment.

Recommendation:

Use constant when the variable's state does not change.

LOWEST	✔ Resolved

Sending back excess ETH

CryptoBearWatchClub.sol::Line 153.
Functionality of public sale returns user excess ETH if he sent more than needed. But pre-sale does not return excess ETH. Even though there is a functionality that allows the owner to collect excess ETH it must be considered.

Recommendation:

Check if excess ETH must NOT be sent back on pre-sale.

LOWEST	✔ Resolved

Magic numbers

CryptoBearWatchClub.sol::Lines 126, 127, 140, 141.
Better try to avoid using magic numbers and declare them as constants to make the code more readable.

Recommendation:

Avoid using magic numbers declaring them as constants.

LOWEST	✔ Unresolved

Naming issue

Arkouda.sol::Lines 99
The functionality requires rewardPaused to be true, in the other case it will revert with error “Reward claiming is turned off” that is a logic issue.

Recommendation:

Replace “rewardPaused” with “!rewardPaused”

	CryptoBearWatchClub	Arkouda
✔ Re-entrancy	Pass	Pass
✔ Access Management Hierarchy	Pass	Pass
✔ Arithmetic Over/Under Flows	Pass	Pass
✔ Delegatecall Unexpected Ether	Pass	Pass
✔ Default Public Visibility	Pass	Pass
✔ Hidden Malicious Code	Pass	Pass
✔ Entropy Illusion (Lack of Randomness)	Pass	Pass
✔ External Contract Referencing	Pass	Pass
✔ Short Address/ Parameter Attack	Pass	Pass
✔ Unchecked CALL Return Values	Pass	Pass
✔ Race Conditions / Front Running	Pass	Pass
✔ General Denial Of Service (DOS)	Pass	Pass
✔ Uninitialized Storage Pointers	Pass	Pass
✔ Floating Points and Precision	Pass	Pass
✔ Tx.Origin Authentication	Pass	Pass
✔ Signatures Replay	Pass	Pass
✔ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Contract: CryptoBearWatchClub

Public mint

- ✓ mint NFT right after auction has started with startingPrice 3 ETH (MAX PRICE) (1068ms)
- ✓ mints NFT after 4500 seconds for a half price (975ms)
- ✓ mints NFT after 9000 seconds for a minimum price (1252ms)
- ✓ doesn't start sale when sale is not open (1041ms)
- ✓ doesn't mint 0 NFT (192ms)
- ✓ doesn't mint when sale is not open (175ms)
- ✓ doesn't mint more than 3 NFT per mint (724ms)
- ✓ doesn't mint when sufficient ETH is not sent (435ms)
- ✓ doesn't mint more than maximum supply of Crypto Bear Watch Club (475ms)

Presale

- ✓ mints NFT for addresses from white list (merkle tree) (874ms)
- ✓ doesn't allow to start presale when presale is not open (108ms)
- ✓ mint NFT for addresses from white list (merkle tree) after 9000 seconds with minimal price (859ms)
- ✓ doesn't premint without merkle tree (113ms)
- ✓ doesn't premint when presale is not open (253ms)
- ✓ doesn't premint when sale is open (684ms)
- ✓ doesn't premint more tokens per transaction than maxMintAllowedPresale (697ms)
- ✓ doesn't allow to premint when sufficient ETH amount not sent (554ms)

Transfer

- ✓ transfer NFT (517ms)
- ✓ safe transfer NFT (469ms)
- ✓ withdrawAll (817ms)

Contract: Arkouda

- ✓ changes reward allowance flag (127ms)
- ✓ sets start time for reward generation (154ms)
- ✓ sets reward amount for tokens by id (211ms)
- ✓ sets allowed addresses for burning tokens (201ms)
- ✓ claims available reward (1451ms)
- ✓ claims no reward if sufficient time hasn't passed (1150ms)
- ✓ claims no reward when reward generation hasn't started (1262ms)
- ✓ gets total claimable amount (894ms)
- ✓ doesn't call updateReward() from not CBWC contract (116ms)
- ✓ burns tokens from allowed accounts (1740ms)
- ✓ doesn't update reward when NFT sent to zero address (625ms)

31 passing (23s)

TEST COVERAGE RESULTS

FILE	% STMTS	% BRANCH	% FUNCS
Arkouda.sol	97.62	85	100
CryptoBearWatchClub.sol	98.61	86.96	95
All files	**98.11**	**85.98**	**97.5**

DISCLAIMER

The information presented in this report is an intellectual property of the customer including all presented documentation, code databases, labels, titles, ways of usage as well as the information about potential vulnerabilities and methods of their exploitation. This audit report does not give any warranties on the absolute security of the code. Blaize.Security is not responsible for how you use this product and does not constitute any investment advice.

Blaize.Security does not provide any warranty that the working product will be compatible with any software, system, protocol or service and operate without interruption. We do not claim the investigated product is able to meet your or anyone else requirements and be fully secure, complete, accurate and free of any errors and code inconsistency.

We are not responsible for all subsequent changes, deletions and relocations of the code within the contracts that are the subjects of this report.

You should perceive Blaize.Security as a tool which helps to investigate and detect the weaknesses and vulnerable parts that may accelerate the technology improvements and faster error elimination.