auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

OptionPrediction

DATED : 1 September 23'

AUDIT SUMMARY

Project name – OptionPrediction

Date: 1 September 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	1
Acknowledged	0	0	2	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Token Information

Proxy Address:
0xf01021CE33586a465aD2723b5d27752054d8e863

Implementation Address:
0xc476b351733ACDbC77eC73633bC8f9E52F4c7460

Network: BSC

Contract Type: P2E

Deployer:
0xEE6fbEC777B8d04423B7964a984E42fCC22e100a

Checksum:
481a8c4dcb4665feeac96a69412e38db5afd3ae8

Test version/Environment:
The tests were performed using forge (foundry).

TOKEN OVERVIEW

OptionPrediction is a platform where players are able to bet on correct answer of a question which is introduced by owner of the platform. Winners will be paid by a portion of total amount of tokens deposited by losers (losers = players who did bet on wrong answer).

Upgradeable platform:

OptionPrediction platform is upgradeable. Meaning that owner is able to update current implementation of the platform.

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

auditace.tech

VULNERABILITY CHECKLIST

- Return values of low-level calls
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

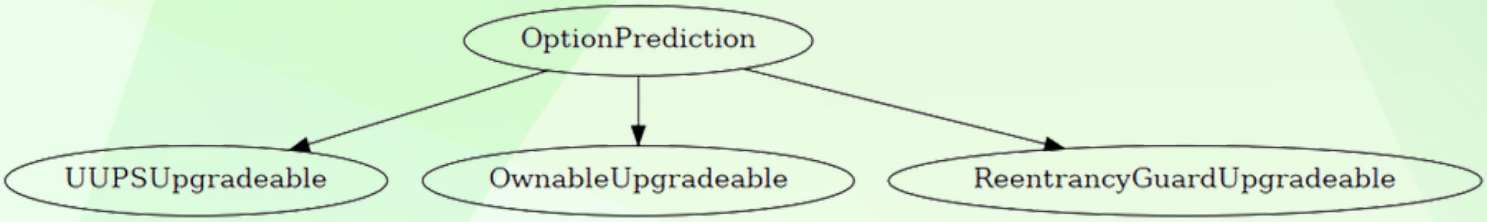
CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	0
Medium-Risk	2
Low-Risk	0
Gas Optimization / Suggestions	1

INHERITANCE TREE



CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
OptionPrediction	Implementation	UUPSUpgradeable, OwnableUpgradeable, ReentrancyGuardUpgradeable		
└	_authorizeUpgrade	Internal 🔒	🛑	onlyOwner
└	initialize	Public ❗	🛑	initializer
└	setAdmin	Public ❗	🛑	onlyOwner nonReentrant
└	setPlatformAddress	Public ❗	🛑	onlyOwner nonReentrant
└	enableCategory	Public ❗	🛑	onlyOwner nonReentrant
└	createNewCategory	Public ❗	🛑	onlyOwner nonReentrant
└	createNewQuestion	Public ❗	🛑	onlyAdmin nonReentrant
└	buyPrediction	Public ❗	🛑	NO❗
└	updateQuestionExpiredTime	Public ❗	🛑	onlyAdmin nonReentrant
└	setQuestionResult	Public ❗	🛑	onlyAdmin nonReentrant
└	deleteQuestion	Public ❗	🛑	onlyOwner nonReentrant
└	claim	Public ❗	🛑	nonReentrant
└	getUserPrediction	Public ❗		NO❗
└	getQuestion	Public ❗		NO❗
└	getQuestionParticipate	Public ❗		NO❗
└	getFee	Public ❗		NO❗
└	getCategory	Public ❗		NO❗
└	getReward	Public ❗		NO❗
└	updateFee	Public ❗	🛑	onlyOwner nonReentrant
└	clearStuckToken	Public ❗	🛑	onlyOwner nonReentrant
└	clearStuckBNB	External ❗	🛑	onlyOwner nonReentrant

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

MANUAL TESTING

Logical – Result of already ended question can be changed

Severity: Medium
functions: setQuestionResult
Status: Acknowledged
Overview:
Result of an already ended (claimed) question can be changed. This may lead to inconsistency between total tokens in the contract and total prediction amounts since new winners are able to claim their tokens which was previously withdrawn by old winners.

```
function setQuestionResult(
address _token,
uint questionId,
uint result
  ) public onlyAdmin nonReentrant {
Category storage category = categories[_token];
require(category.token == _token, CATEGORY_NOTFOUND);

Question storage question = category.questionMap[questionId];
require(question.questionId == questionId, QUESTION_NOTFOUND);
require(!question.isDeleted, QUESTION_NOTFOUND);

require(
question.expiredTime < block.timestamp,
ERROR_GAME_STILL_NOT_START
);
require(result < 2, ERROR_INVALID_RESULT);
question.bullseye = result;

emit ResultSet(_token, questionId, result);
  }
```

Suggestion
Ensure that result of already claimed questions can not be changed later.

MANUAL TESTING

Logical – Already ended question can be deleted

Severity: Medium
functions: setQuestionResult
Status: Acknowledged
Overview:
An already ended (claimed) question can be deleted. This may lead to inconsistency between total tokens in the contract and total prediction amounts since participates will receive total amount of prediction they have made before.

```
function deleteQuestion(
address _token,
uint questionId
  ) public onlyOwner nonReentrant {
Category storage category = categories[_token];
require(category.token == _token, CATEGORY_NOTFOUND);

Question storage question = category.questionMap[questionId];
require(question.questionId == questionId, QUESTION_NOTFOUND);
require(!question.isDeleted, QUESTION_NOTFOUND);
question.isDeleted = true;
//@audit predict info of participates are not reset to 0
for (uint256 i = 0; i < question.participates.length; i++) {
address player = question.participates[i];
UserPrediction storage predictionInfo = question.participate[
        player
];
if (
predictionInfo.gate0TotalPrediction > 0 ||
        predictionInfo.gate1TotalPrediction > 0
) {
        ERC20(_token).transfer(
           player,
predictionInfo.gate0TotalPrediction.add(
predictionInfo.gate1TotalPrediction
           )
        );
}
}
emit QuestionDeleted(_token, questionId);
  }
```

Suggestion
Ensure that a question can not be deleted after it has been ended.
Alleviation:
it might be essential to delete the question just created by mistake. We will refund all the users who have bet on this question.

MANUAL TESTING

Events – Lack of “indexed” keyword

Severity: Informational

functions: ---

Status: Open

Overview:

Events are not using “indexed” keywords. “indexed” keyword makes event queries faster as results can be filtered by a specifiec field (e.g. only getting questions from token “X”)

Suggestion

Make sure to use “indexed” keyword. Indexed keyword can be used 3 times in each event.

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace