OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 18	2022-03-07	DoD4uFN, Donut	Audit Draft

Audit Notes

Audit Date	2022-02-19 - 2022-03-06
Auditor/Auditors	DoD4uFN, ByFixter
Auditor/Auditors Contact Information	contact@obeliskauditing.com
Notes	Specified code and contracts are audited for security flaws. UI/UX (website), logic, team, and tokenomics are not audited.
Audit Report Number	OB566535849

Disclaimer

This audit is not financial, investment, or any other kind of advice and is for informational purposes only. This report is not a substitute for doing your own research and due diligence. Obelisk is not responsible or liable for any loss, damage, or otherwise caused by reliance on this report for any purpose. Obelisk has based this audit report solely on the information provided by the audited party and on facts that existed before or during the audit being conducted. Obelisk is not responsible for any outcome, including changes done to the contract/contracts after the audit was published. This audit is fully objective and only discerns what the contract is saying without adding any opinion to it. The audit is paid by the project but neither the auditors nor Obelisk has any other connection to the project and has no obligations other than to publish an objective report. Obelisk will always publish its findings regardless of the outcome of the findings. The audit only covers the subject areas detailed in this report and unless specifically stated, nothing else has been audited. Obelisk assumes that the provided information and material were not altered, suppressed, or misleading. This report is published by Obelisk, and Obelisk has sole ownership of this report. Use of this report for any reason other than for informational purposes on the subjects reviewed in this report including the use of any part of this report is prohibited without the express written consent of Obelisk. In instances where an auditor or team member has a personal connection with the audited project, that auditor or team member will be excluded from viewing or impacting any internal communication regarding the specific audit.

Obelisk Auditing

Defi is a relatively new concept but has seen exponential growth to a point where there is a multitude of new projects created every day. In a fast-paced world like this, there will also be an enormous amount of scams. The scams have become so elaborate that it's hard for the common investor to trust a project, even though it could be legit. We saw a need for creating high-quality audits at a fast phase to keep up with the constantly expanding market. With the Obelisk stamp of approval, a legitimate project can easily grow its user base exponentially in a world where trust means everything. Obelisk Auditing consists of a group of security experts that specialize in security and structural operations, with previous work experience from among other things, PricewaterhouseCoopers. All our audits will always be conducted by at least two independent auditors for maximum security and professionalism.

As a comprehensive security firm, Obelisk provides all kinds of audits and project assistance.

Audit Information

The auditors always conducted a manual visual inspection of the code to find security flaws that automatic tests would not find. Comprehensive tests are also conducted in a specific test environment that utilizes exact copies of the published contract.

While conducting the audit, the Obelisk security team uses best practices to ensure that the reviewed contracts are thoroughly examined against all angles of attack. This is done by evaluating the codebase and whether it gives rise to significant risks. During the audit, Obelisk assesses the risks and assigns a risk level to each section together with an explanatory comment. Take note that the comments from the project team are their opinion and not the opinion of Obelisk.

Table of Contents

Project Information

Name	Assent Protocol
Description	ΛssentProtocol is building a community-owned decentralized financially secure infrastructure to bring more stability and transparency to investors.
Website	https://t.co/mdfQklyo61
Contact	https://twitter.com/Assent_Protocol
Contact information	@AssentProtocol on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	BSC

Audit of Assent Protocol

> **No serious issues were found, and those informational findings that were found were swiftly fixed.**

Obelisk was commissioned by Assent Protocol on the 15th of February 2022 to conduct a comprehensive audit of Assent Protocols' contracts. The following audit was conducted between the 19th of February 2022 and the 6th of March 2022. Two of Obelisk's security experts went through the related contracts manually using industry standards to find if any vulnerabilities could be exploited either by the project team or users.

The informational findings are good to know while interacting with the project but don't directly damage the project in its current state, hence it's up to the project team if they deem that it's worth solving these issues.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the Assent Protocol project**.** This document is a summary of the findings that the auditors found.

Please read the full document for a complete understanding of the audit.

Summary Table

Finding	ID	Severity	Status
Protocol Trading Fees Distribution	#0001	Informational	Closed
Unused Function	#0002	Informational	Closed
Require Statement Message	#0003	Informational	Closed
Missing Zero Check	#0004	Informational	Closed
No Events Emitted For Changes To Protocol Values	#0005	Informational	Closed

Findings

Manual Analysis

Protocol Trading Fees Distribution

FINDING ID	#0001
SEVERITY	Informational
STATUS	Closed
LOCATION	AssentPair.sol -> 112-130

```
1     function _mintFee(uint112 _reserve0, uint112 _reserve1) private
  returns (bool feeOn) {
2         address feeTo = IAssentFactory(factory).feeTo();
3         feeOn = feeTo != address(0);
4         uint _kLast = kLast; // gas savings
5         if (feeOn) {
6             if (_kLast != 0) {
7                 uint rootK = Math.sqrt(uint(_reserve0).mul(_reserve1));
8                 uint rootKLast = Math.sqrt(_kLast);
9                 if (rootK > rootKLast) {
10                    uint numerator =
  totalSupply.mul(rootK.sub(rootKLast));
11                    uint denominator = (rootK / 3).add(rootKLast);
12                    uint liquidity = numerator / denominator;
13                    if (liquidity > 0) _mint(feeTo, liquidity);
14                }
15            }
16        } else if (_kLast != 0) {
17            kLast = 0;
18        }
19    }
```

DESCRIPTION	The distribution of the protocol's trading fees is 25% to the liquidity providers and 75% to the *feeTo* address.
RECOMMENDATION	Verify that this is the intended behavior.
RESOLUTION	Project team comment: “Yes, it's intended to have this protocol's trading fees repartition to collect them into the protocol and redistribute them to native token holders. The value is written into the function comment on (previous) line 111.”

Unused Function

FINDING ID	#0002
SEVERITY	Informational
STATUS	Closed
LOCATION	• AssentLibrary.sol -> 35-37: *function getSwapFee(address factory, address tokenA, address tokenB) internal view returns (uint swapFee)*

DESCRIPTION	The function is not being used internally by the protocol.
RECOMMENDATION	Remove the redundant function.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 0524cd5a33e66cfe2f34ce20e713d3fc03261263

Require Statement Message

FINDING ID	#0003
SEVERITY	Informational
STATUS	Closed
LOCATION	• AssentPair.sol -> 88: *require(_swapFee > 0, "AssentSwap: lower than 0");*

DESCRIPTION	The argument *_swapFee* is *uint32* which cannot be lower than zero. The functionality of the statement is correct.
RECOMMENDATION	Change the message to reflect the statement.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 0524cd5a33e66cfe2f34ce20e713d3fc03261263

Static Analysis

Missing Zero Check

FINDING ID	#0004
SEVERITY	Informational
STATUS	Closed
LOCATION	AssentFactory.sol -> 30-33

```
1    constructor(address _feeToSetter,address _feeTo) public {
2        feeTo = _feeTo;
3        feeToSetter = _feeToSetter;
4    }
```

DESCRIPTION	Contract address values in the constructor can be set to zero address. Zero addresses may cause incorrect contract behavior.
RECOMMENDATION	Add a check to ensure contract values are never set to an invalid zero address.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 0524cd5a33e66cfe2f34ce20e713d3fc03261263

No Events Emitted For Changes To Protocol Values

FINDING ID	#0005
SEVERITY	Informational
STATUS	Closed
LOCATION	• AssentFactory.sol -> 73-78: *function setWhitelist(IAssentWhitelist _whitelist) external*

DESCRIPTION	Functions that change important variables should emit events such that users can more easily monitor the change.
RECOMMENDATION	Emit events from these functions.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 0524cd5a33e66cfe2f34ce20e713d3fc03261263

On-Chain Analysis

No Relevant Findings

External Contracts

These contracts are not part of the audit scope.

Deposit Tokens

ADDRESS	ETH 0x2170Ed0880ac9A755fd29B2688956BD959F933F8 WBNB 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
USAGE	0x036Db024EcD69C142E20d7edeaeE90ffC4A34Fbf *AssentPair.token0* - Constant *AssentPair.token1* - Constant
IMPACT	• ERC20 Token

Appendix A - Reviewed Documents

Document	Address
AssentERC20.sol	N/A
AssentFactory.sol	0x5B3C1F260E09e653290f24F75abC5e466fD42310
AssentLibrary.sol	N/A
AssentPair.sol	AssentLP ETH-WBNB 0x036Db024EcD69C142E20d7edeaeE90ffC4A34Fbf
AssentRouter.sol	0x2Df0d214239E20535060220aE54ef361606e346b
IAssentCallee.sol	N/A
IAssentERC20.sol	N/A
IAssentFactory.sol	N/A
IAssentPair.sol	N/A
IAssentRouter01.sol	N/A
IAssentRouter02.sol	N/A
IAssentWhitelist.sol	N/A
IERC20.sol	N/A
IWETH.sol	N/A
Math.sol	N/A
SafeMath.sol	N/A
TransferHelper.sol	N/A
UQ112x112.sol	N/A

Revisions

Revision 1	75a89019e53a8976a75a081d09b30a3aae9f462a
Revision 2	0524cd5a33e66cfe2f34ce20e713d3fc03261263

Imported Contracts

No imported contracts.

Appendix B - Risk Ratings

Risk	Description
High Risk	A fatal vulnerability that can cause the loss of all Tokens / Funds.
Medium Risk	A vulnerability that can cause the loss of some Tokens / Funds.
Low Risk	A vulnerability that can cause the loss of protocol functionality.
Informational	Non-security issues such as functionality, style, and convention.

Appendix C - Finding Statuses

Closed	Contracts were modified to permanently resolve the finding.
Mitigated	The finding was resolved by other methods such as revoking contract ownership. The issue may require monitoring, for example in the case of a time lock.
Partially Closed	Contracts were updated to fix the issue in some parts of the code.
Partially Mitigated	Fixed by project-specific methods which cannot be verified on-chain. Examples include compounding at a given frequency.
Open	The finding was not addressed.

Appendix D - Audit Procedure

A typical Obelisk audit uses a combination of the three following methods:

Manual analysis consists of a direct inspection of the contracts to identify any security issues. Obelisk auditors use their experience in software development to spot vulnerabilities. Their familiarity with common contracts allows them to identify a wide range of issues in both forked contracts as well as original code.

Static analysis is software analysis of the contracts. Such analysis is called “static” as it examines the code outside of a runtime environment. Static analysis is a powerful tool used by auditors to identify subtle issues and to verify the results of manual analysis.

On-chain analysis is the audit of the contracts as they are deployed on the blockchain. This procedure verifies that:

- deployed contracts match those which were audited in manual/static analysis;
- contract values are set to reasonable values;
- contracts are connected so that interdependent contracts function correctly;
- and the ability to modify contract values is restricted via a timelock or DAO mechanism. (We recommend a timelock value of at least 72 hours)

Each obelisk audit is performed by at least two independent auditors who perform their analysis separately.

After the analysis is complete, the auditors will make recommendations for each issue based on best practices and industry standards. The project team can then resolve the issues, and the auditors will verify that the issues have been resolved with no new issues introduced.

Our auditing method lays a particular focus on the following important concepts:

- Quality code and the use of best practices, industry standards, and thoroughly tested libraries.
- Testing the contract from different angles to ensure that it works under a multitude of circumstances.
- Referencing the contracts through databases of common security flaws.

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group