



Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 22	2021-09-11	Zapmore, MrTeaThyme	Audit Final

Audit Notes

Audit Date	2021-08-02 - 2021-09-09
Auditor/Auditors	MrTeaThyme, DoD4uFN
Auditor/Auditors Contact Information	tibereum-obelisk@protonmail.com
Notes	Specified code and contracts are audited for security flaws. UI/UX (website), logic, team, and tokenomics are not audited.
Audit Report Number	OB555678912

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.

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.

Table of Content

Version Notes	2
Audit Notes	2
Disclaimer	2
Obelisk Auditing	3
Project Information	5
Executive Summary	6
Summary Table	7
Introduction	8
Findings	9
Manual Analysis	9
No Lower Bound On numPeriods	9
RefOwner Can Withdraw Funds From Vesting Bucket Without Checks	11
No Events Emitted To Important Changes	12
Missing Zero Checks	14
Expensive Loop	15
Static Analysis	16
No Relevant Findings	16
On-Chain Analysis	17
No Relevant Findings	17
Appendix A - Reviewed Documents	18
Appendix B - Risk Ratings	20
Appendix C - Icons	20
Appendix D - Testing Standard	21

Project Information

Project Name	VegaSwap
Description	Vegaswap is an AMM built with multi-chain in mind enabling a wide range of Defi and Cross-chain applications. With SMART pools anyone can become a marketmaker and build wealth by providing liquidity.
Website	https://vegaswap.io/
Contact	@benjyz#7236
Contact information	@benjyz#7236 on Discord
Token Name(s)	vega
Token Short	vega
Contract(s)	See Appendix A
Code Language	Solidity/Vyper
Chain	Multichain

Executive Summary

The audit of VegaSwap's token, vesting contract, and presale contract was conducted by two of Obelisks' security experts between the 2nd of July 2021 and the 9th of September 2021.

After finishing the full audit, Obelisk can safely state that all relevant findings were mitigated by the project team.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the VegaSwap project.

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

Summary Table

Audited Part	ID	Severity	Mitigated
No Lower Bound On numPeriods	#0001	Low Risk	Resolved
RefOwner Can Withdraw Funds From Vesting Bucket Without Checks	#0002	Low Risk	Resolved
No Events Emitted To Important Changes	#0003	Informational	Resolved
Missing Zero Checks	#0004	Informational	Resolved
Expensive Loop	#0005	Informational	Resolved

Introduction

Obelisk was commissioned by VegaSwap on the 26th of July 2021 to conduct a comprehensive audit of VegaSwap's *token*, the *vesting contract*, and the *presale contract*. The following audit was conducted between the 2nd of August 2021 and the 9th of September 2021 and delivered on the 11th of September 2021. Two of Obelisk's security experts went through the related contracts using industry standards to find if any vulnerabilities could be exploited.

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

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.

The audit was conducted on contracts that were not yet live in a production environment. A comprehensive on-chain analysis was conducted as the contracts were deployed in order to match the audited contracts with the published contracts.

During the initial audit, we found 2 Low Severity issues and 3 informational issues. These issues were presented to the project team. The project team decided to rewrite the contracts using Vyper and while doing that solved the issues found. Obelisk then re-audited the newly rewritten Vyper contracts and found no new issues and that all the presented issues were resolved.

The informational findings are good to know while interacting with the project but don't directly damage the project in its current state.

Please see each section of the audit to get a full understanding of the audit.

Findings

Manual Analysis

No Lower Bound On numPeriods

SEVERITY	Low Risk
RESOLVED	YES
FINDING ID	#0001
LOCATION	VestingBucket.sol -> 44-67

```
1  constructor(  
2      address _VEGA_TOKEN_ADDRESS,  
3      uint256 _cliffTime,  
4      uint256 _numPeriods,  
5      uint256 _totalAmount  
6  ) AbstractBucket(_VEGA_TOKEN_ADDRESS) {  
7      require(  
8          _cliffTime >= block.timestamp,  
9          "VESTINGBUCKET cliff must be in the future"  
10     );  
11     cliffTime = _cliffTime;  
12     numPeriods = _numPeriods;  
13     totalAmount = _totalAmount;  
14  
15     bucketAmountPerPeriod = totalAmount / numPeriods;  
16     endTime = VestingMath.getEndTime(  
17         _cliffTime,  
18         bucketAmountPerPeriod,  
19         _totalAmount  
20     );  
21  
22     totalWithdrawnAmount = 0;  
23     totalClaimAmount = 0;  
24     //claimAddresses =  
25 }
```

DESCRIPTION

Since *numPeriods* is used for vesting, it would be good practice to have a lower bound, so that it cannot be set to 0 (in which case it won't be able to be called at all).

RECOMMENDATION	Add checks for <i>numberPeriods</i> . Consider adding checks for <i>_cliffTime</i> and <i>_totalAmount</i> if deemed necessary.
MITIGATED/COMMENT	Mitigated during a refactor to Vyper.

RefOwner Can Withdraw Funds From Vesting Bucket Without Checks

SEVERITY	Low Risk
RESOLVED	YES
FINDING ID	#0002
LOCATION	VestingBucket.sol -> 175-180

```
1  function withdrawOwner(uint256 amount) public onlyRefOwner {  
2      //check existing claims  
3      bool transferSuccess = vega_token.transfer(msg.sender, amount);  
4      require(transferSuccess, "VESTINGBUCKET: withdrawOwner failed");  
5      emit WithdrawOwner(amount);  
6  }
```

DESCRIPTION	<i>RefOwner</i> is able to withdraw all the funds from a Vesting Bucket without limitations.
RECOMMENDATION	Checks should be added to ensure the safety of the funds in the Vesting Bucket.
MITIGATED/COMMENT	Mitigated during a refactor to Vyper.

No Events Emitted To Important Changes

SEVERITY	Informational
RESOLVED	YES
FINDING ID	#0003
LOCATION	RefOwnable.sol -> 18-20 VestingBucket.sol -> 70-99 VegaMaster.sol -> 31-51



```
1 function setRefOwner(address _refowner) public onlyOwner {  
2     _refOwner = _refowner;  
3 }
```



```
1 function addClaim(address _claimAddress, uint256 _claimTotalAmount)  
2 public  
3 onlyRefOwner  
4 {  
5     ...  
6 }
```



```
1    function addVestingBucket(  
2        uint256 cliffOffset,  
3        string memory name,  
4        uint256 periods,  
5        uint256 amount  
6    ) public onlyOwner {  
7        ...  
8    }
```

DESCRIPTION

There is no event emit when setting the refOwner. Since refOwner is a privileged address, it should emit an event. When an address is added to claim the vested funds, an event should be emitted including the address, period, amount. An event should be emitted when a vesting Bucket is created including cliffOffset, periods, amount.

RECOMMENDATION

Add events to the aforementioned functions.

MITIGATED/COMMENT

Mitigated during a refactor to Vyper.

Missing Zero Checks

SEVERITY	Informational
RESOLVED	YES
FINDING ID	#0004
LOCATION	AbstractBucket.sol -> 14-17: constructor(address _VEGA_TOKEN_ADDRESS) VegalDO.sol -> 31-44: constructor(address _launchTokenAddress, address _investTokenAddress, uint256 _askPriceMultiple, uint256 _cap)
DESCRIPTION	Constructor doesn't check for zero address before assigning to variables.
RECOMMENDATION	Add a check for zero address if deemed necessary.
MITIGATED/COMMENT	Mitigated during a refactor to Vyper.

Expensive Loop

SEVERITY	Informational
RESOLVED	YES
FINDING ID	#0005
LOCATION	VegalDO.sol -> 62-70

```
1  function inWhitelist(address f) public view returns (bool) {
2      for (uint256 i = 0; i < whitelistAddresses.length; i++) {
3          address w = whitelistAddresses[i];
4          if (w == f) {
5              return true;
6          }
7      }
8      return false;
9  }
```

DESCRIPTION	The function <i>inWhitelist</i> iterates over the <i>whitelistAddresses</i> array in order to check the existence of a user inside it. This approach is both potentially expensive and slow.
RECOMMENDATION	Use mapping to track whether a given address is in <i>whitelistAddresses</i> .
MITIGATED/COMMENT	Mitigated during a refactor to Vyper.

Static Analysis

No Relevant Findings

On-Chain Analysis

No Relevant Findings

Appendix A - Reviewed Documents




Document	Address
AbstractBucket.sol	N/A
MaxSupplyToken.sol	N/A
Ownable.sol	N/A
RefOwnable.sol	N/A
VegalDO.sol	N/A
VegaMaster.sol	N/A
VegaToken.sol	N/A
VestingBucket.sol	N/A
IERC20.sol	N/A
VestingConstants.sol	N/A
VestingMath.sol	N/A
Bucket.vy	<p>Advisors: '0x4d91BaCD3F2CC3BCB18F0027381C7F1c44363C88'</p> <p>Development: '0x1dAA4943280C85be711A6bDbACb7b73c66601B6d'</p> <p>Ecosystem: '0x45f96c234140350182E33bDc83A649C27feF8369'</p> <p>LP_grants: '0x680052c12D864f7FD61673261453dE63732Fb102'</p> <p>LP_rewards: '0x13C4168962A3c1DFe02f8eD9591d8EDd28f3E047'</p> <p>Marketing: '0xed223E325006699418b9965458cB2c6D5bd08533'</p> <p>Private: '0x4464B3917d9A249ff99D8Aa89E5a35D373Af2305'</p> <p>Public_Vested: '0xB4df471B6509a1078a799a4f49CaB97e719d7Ab5'</p> <p>Seed: '0xAD558B957010B6C75463E883B109Dc9322Faa06f'</p> <p>Team: '0x3e632E8A689407a397f5713c6A581CC56410Ed7E'</p> <p>Trade_Mining: '0xD7a8B70c101A03b59d160abC18205191c9415070'</p> <p>Treasury:</p>

	'0x69BC0fB381c0E5b030D72b33C3C311BCb6d8ceD5' Vega_Liquidity: '0x3A486587c0f5dbf18166FA7e470AfC9c2Ba73555'
ClaimList.vy	N/A
VegaToken.vy	0x4EfDFe8fFAfF109451Fc306e0B529B088597dd8d

Appendix B - Risk Ratings

Risk	Description
High Risk	A fatal vulnerability that can cause immediate loss of Tokens / Funds
Medium Risk	A vulnerability that can cause some loss of Tokens / Funds
Low Risk	A vulnerability that can be mitigated
Informational	No vulnerability

Appendix C - Icons

Icon	Explanation
	Solved by Project Team
	Under Investigation of Project Team
	Unsolved

Appendix D - Testing Standard

An ordinary audit is conducted using these steps.

1. Gather all information
2. Conduct a first visual inspection of documents and contracts
3. Go through all functions of the contract manually (2 independent auditors)
 - a. Discuss findings
4. Use specialized tools to find security flaws
 - a. Discuss findings
5. Follow up with project lead of findings
6. If there are flaws, and they are corrected, restart from step 2
7. Write and publish a report

During our audit, a thorough investigation has been conducted employing both automated analysis and manual inspection techniques. Our auditing method lays a particular focus on the following important concepts:

- Ensuring that the code and codebase use best practices, industry standards, and available libraries.
- Testing the contract from different angles ensuring that it works under a multitude of circumstances.
- Analyzing the contracts through databases of common security flaws.

Follow Obelisk Auditing for the Latest Information



ObeliskOrg



ObeliskOrg



Part of Tibereum Group