OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 20	2022-01-13	Zapmore, DoD4uFN	Audit Final

Audit Notes

Audit Date	2022-01-02 - 2022-02-07
Auditor/Auditors	DoD4uFN, thing_theory
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	OB556326326

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.

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	Unilab
Description	Unilab is built to solve the cross-cutting concerns across the life-cycle of smart contract development, deployment & management for everyone.
Website	https://unilab.network/
Contact	@Geott1 on TG
Contact information	@Geott1 on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Polygon / BSC

Audit of Unilab

> **Mostly informational findings with a complexity added to see excluded addresses, and with a BOT protection that need to be disabled for full security.**

Obelisk was commissioned by Unilab on the 30th of December 2021 to conduct a comprehensive audit of Unilabs' Token contracts. The following audit was conducted between the 2nd of January 2022 and the 2nd of February 2021. 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.

During the audit, the auditors mostly found informational findings which didn't impact the project in any major way. There were a low risk issue finding, #1, which is acknowledged and commented on by the team. It's an issue that refers to its complexity to see which addresses can be excluded. Issue #7 is also a low risk finding where the contract is owned by an EOA address which is recommended to be put behind a 72h timelock.

We do have a high risk finding and that is issue #6. Issue #6 is acknowledged by the project team, and stems from their IDO where the BOT protection is supposed to help against BOTS and manage whitelists. The project team states that the BOT protection will be disabled a few days after listing, so this issue is present until it's disabled. Note that this BOT protection is not audited by OBELISK

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 Unilab project**.**

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

Summary Table

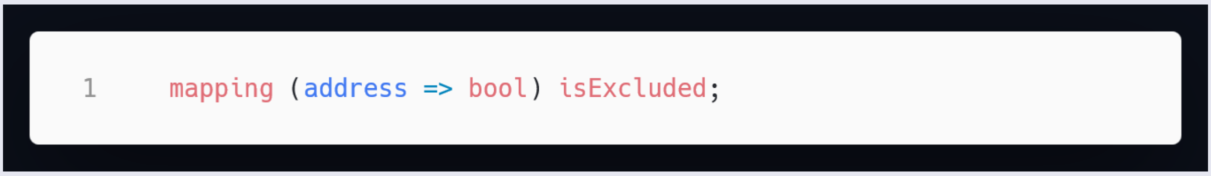
Finding	ID	Severity	Status
Addresses Excluded From Swap Tax Cannot Easily Be Identified	#0001	Low Risk	Open
Wrong Event Emitted	#0002	Informational	Closed
Make Use Of Protocol's Variables	#0003	Informational	Closed
Missing Zero Checks	#0004	Informational	Mitigated
Write without Read	#0005	Informational	Closed
Transfers And Swaps Can Be Disabled	#0006	High Risk	Partially Mitigated
Owner Is An EOA	#0007	Low Risk	Open
Bot Protection Not Disabled Forever	#0008	Informational	Open

Findings

Manual Analysis

Addresses Excluded From Swap Tax Cannot Easily Be Identified

FINDING ID	#0001
SEVERITY	Low Risk
STATUS	Open
LOCATION	[ERC20SupportingFeesInQuoteToken.sol -> 21]()

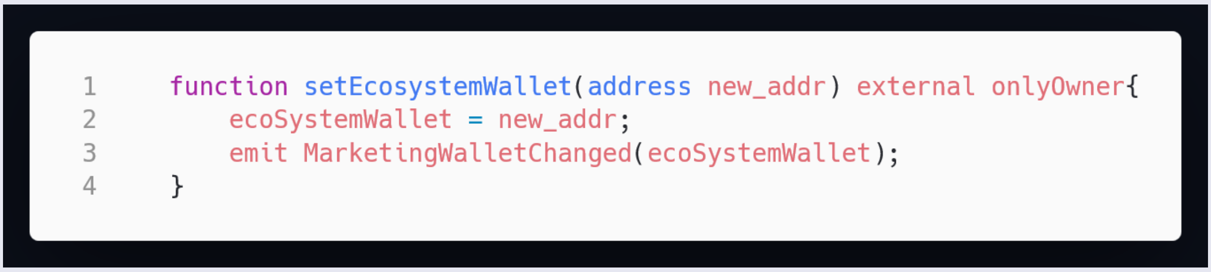


```solidity
mapping (address => bool) isExcluded;
```

DESCRIPTION	The addresses contained in *isExcluded* are not easily identified by users. It is important to ensure that all addresses are accounted for.
RECOMMENDATION	Add an enumerated list of all *isExcluded* addresses. Add events that emit when an address is added or removed.
RESOLUTION	*isExcluded* was made public, but there is still no way to easily iterate through all excluded addresses. Team comment: *acknowledged, it should be visible using events so we will not change it for now* Reviewed in commit 0db13f0964e0ed6a8a25714dd872c7d497f1c788

Wrong Event Emitted

FINDING ID	#0002
SEVERITY	Informational
STATUS	Closed
LOCATION	[ERC20SupportingFeesInQuoteToken.sol -> 91-94]()

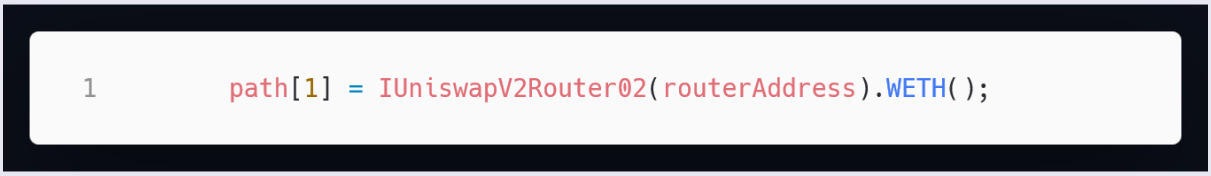


```
function setEcosystemWallet(address new_addr) external onlyOwner{
    ecoSystemWallet = new_addr;
    emit MarketingWalletChanged(ecoSystemWallet);
}
```

DESCRIPTION	Function *setEcosystemWallet()* emits event *MarketingWalletChanged()*.
RECOMMENDATION	Change the event to *EcosystemWalletChanged()*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 0db13f0964e0ed6a8a25714dd872c7d497f1c788

Make Use Of Protocol's Variables

FINDING ID	#0003
SEVERITY	Informational
STATUS	Closed
LOCATION	[ERC20SupportingFeesInQuoteToken.sol -> 157]()

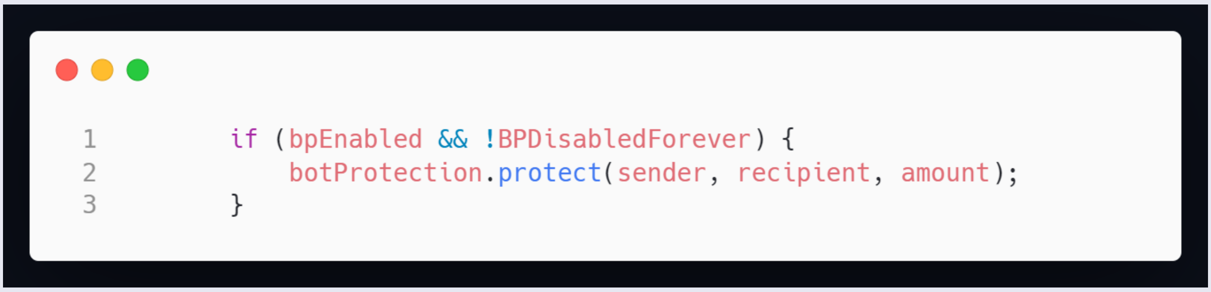


```
        path[1] = IUniswapV2Router02(routerAddress).WETH();
```

DESCRIPTION	The *routerAddress* is being wrapped to *IUniswapV2Router02* to call *.WETH()*.
RECOMMENDATION	Make use of the *router* variable which is already wrapped.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 0db13f0964e0ed6a8a25714dd872c7d497f1c788

Transfers And Swaps Can Be Disabled

FINDING ID	#0006
SEVERITY	High Risk
STATUS	Partially Mitigated
LOCATION	ERC20SupportingFeesInQuoteToken.sol -> 142-144



```
        if (bpEnabled && !BPDisabledForever) {
            botProtection.protect(sender, recipient, amount);
        }
```

DESCRIPTION	If *botProtection.protect* reverts, it will prevent all transfers and therefore swaps.
RECOMMENDATION	Make sure bot protection doesn't revert except when intended.
RESOLUTION	Issue #6 is acknowledged by the project team, and stems from their IDO where the BOT protection is supposed to help against BOTS and manage whitelists. The project team states that the BOT protection will be disabled a few days after listing, so this issue is present until it's disabled. Note that this BOT protection is NOT audited by OBELISK

Static Analysis

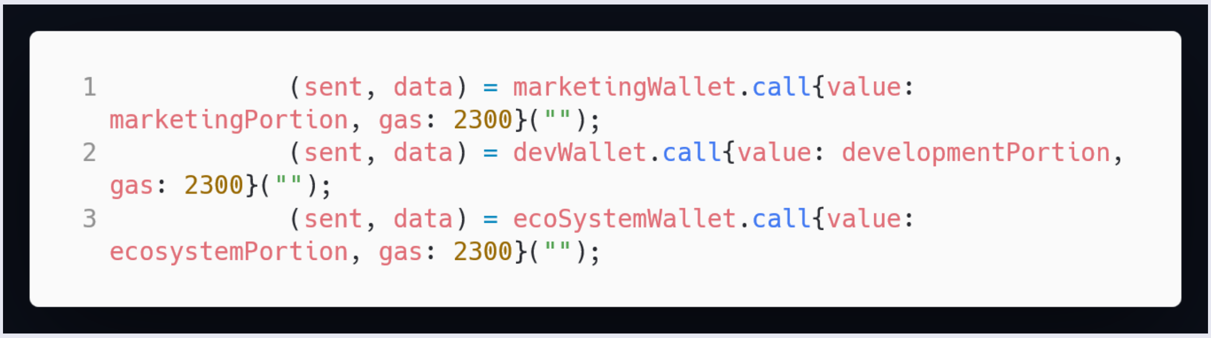
Missing Zero Checks

FINDING ID	#0004
SEVERITY	Informational
STATUS	Mitigated
LOCATION	• ERC20SupportingFeesInQuoteToken.sol -> 48: *baseToken = baseToken_;* • ERC20SupportingFeesInQuoteToken.sol -> 49: *routerAddress = router_;* • ERC20SupportingFeesInQuoteToken.sol -> 81: *function setDevWallet(address new_addr) external onlyOwner{* • ERC20SupportingFeesInQuoteToken.sol -> 86: *function setMarketingWallet(address new_addr) external onlyOwner{* • ERC20SupportingFeesInQuoteToken.sol -> 91: *function setEcosystemWallet(address new_addr) external onlyOwner{*

DESCRIPTION	The contract address values can be set to zero address in various constructors, initializers, and setter functions. 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 won't apply checks for *baseToken* and *routerAddress* because they are set once at deployment. Reviewed in commit 0db13f0964e0ed6a8a25714dd872c7d497f1c788

Write without Read

FINDING ID	#0005
SEVERITY	Informational
STATUS	Closed
LOCATION	[ERC20SupportingFeesInQuoteToken.sol -> 176-178]



```
            (sent, data) = marketingWallet.call{value: marketingPortion, gas: 2300}("");
            (sent, data) = devWallet.call{value: developmentPortion, gas: 2300}("");
            (sent, data) = ecoSystemWallet.call{value: ecosystemPortion, gas: 2300}("");
```

DESCRIPTION	The variables *sent* and *data* are written, but are not read before being written again.
RECOMMENDATION	Remove the variables if they are unneeded.
RESOLUTION	*sent* variable is being used to emit an event. Reviewed in commit 0db13f0964e0ed6a8a25714dd872c7d497f1c788

On-Chain Analysis

Owner Is An EOA

FINDING ID	#0007
SEVERITY	Low Risk
STATUS	Open
LOCATION	Unilab 0x7111E5C9b01ffA18957B1AA27E9Cb0e8FBA214F5

DESCRIPTION	The Owner of the Unilab contract is an EOA. The functions that only the Owner can call are : *setMarketingFee(), setRndFee(), setEcosystemFee(), setRndWallet(), setMarketingWallet(), setEcosystemWallet(), setPairBalanceThreshold(), setSwapForMarketing(), setMaxTokensToSwapForFees(), swapForFees(), multiExcludeFromFees(), multiIncludeInFees(), isExcludedFromFees(), setBPAddress(), setBpEnabled(), setBotProtectionDisableForever()*. Owner 0x0ef9ccF56e6bbDe033848dCAACc243F3A7305B03
RECOMMENDATION	It is recommended to have a timelock of at least 72h.
RESOLUTION	N/A

Bot Protection Not Disabled Forever

FINDING ID	#0008
SEVERITY	Informational
STATUS	Open
LOCATION	Unilab 0x7111E5C9b01ffA18957B1AA27E9Cb0e8FBA214F5

DESCRIPTION	The *bpEnabled* is False. However, the *BPDisabledForever* isn't True yet. Protection can still be enabled at this point, refer to issue #0006 *Transfers And Swaps Can Be Disabled*. The bot protection contract: 0x032eCE6B24C878B24BEF021BB916B1CEBB8324dC
RECOMMENDATION	Disable the bot protection.
RESOLUTION	N/A

Appendix A - Reviewed Documents

Document	**Address**
Unilab.sol	0x7111E5C9b01ffA18957B1AA27E9Cb0e8FBA214F5

Revisions

Revision 1	0bd4601108ec2af4e8120eeb8c1c4925c94bd32e

Imported Contracts

OpenZeppelin	Version
ERC20Burnable	v4.4.1
ERC20	v4.4.1
Context	v4.4.1
Ownable	v4.4.1
IERC20	v4.4.1
IERC20Metadata	v4.4.1

Externally Owned Accounts

Account	Address
owner	0x0ef9ccF56e6bbDe033848dCAACc243F3A7305B03
ecosystemWallet	0xF78BD06f2528996B3a409D382384E4433F8965f9
marketingWallet	0xEF0cAd5B5227867b223C25a6217C5FcCE60fA983
rndWallet	0x398e85b7cD2E4811a7a4548Fe9e4197468Db0543

External Contracts

These contracts are not part of the audit scope.

botProtection	0x032eCE6B24C878B24BEF021BB916B1CEBB8324dC
Pancake Router	0x10ED43C718714eb63d5aA57B78B54704E256024E
WBNB	0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c

WBNB-Unilab Pancake LP	0xCa05Ce5534E26902d2FBa42dDA95DFf389b43143

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 which 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 block-chain. 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 contract 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 practice 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