OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 31	2022-11-22	DoD4uFN, Donut	Audit Final

Audit Notes

Audit Date	2022-10-07 - 2022-11-21
Auditor/Auditors	DoD4uFN, mechwar
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	OB599851241

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	Based
Description	BASED Next Generation protocol is the first pegless seigniorage protocol exploring DeFi on the FTM Network. We are introducing innovative yield strategies whilst providing inclusivity for Based Finance (V1), that successfully finished emissions. Based Next Gen is a multi-token protocol which consists of the following tokens: $OBOL - PEGLESS token with elastic supply. $SMELT - protocol's underlying Perpetual Print (PP) mechanism emitting OBOL.
Website	https://basedfinance.io/
Contact	https://twitter.com/BasedFinance_io
Contact information	@athena_goddazz on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Fantom

Audit of Based-Dex

Obelisk was commissioned by Based on the 5th of October 2022 to conduct a comprehensive audit of Based' Dex contracts. The following audit was conducted between the 7th of October 2022 and the 21st of November 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 reason for the big timespan of the audit was that the contracts were updated during the audit with changes that also needed to be audited.

During the audit, the auditors found one High risk and one medium-risk issue that was fixed before deployment. There were also two Low-risk findings where one was closed by the project team and the other, #3 is still open. The open issue relates to specific instances where a transaction can just revert.

The on-chain issue #7 relates to missing timelock which should be added to increase security.

Issue #8 relates to the contracts not being verified on-chain. Obelisk which has access to the full contracts has used methods in order to verify that the contracts deployed to match the ones audited. However, we still recommend the contracts be visible for everyone to verify.

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, however, please take note of them.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the Based 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

Code Analysis

Finding	ID	Severity	Status
Taxes Can Be Used To Drain User Funds	#0001	High Risk	Closed
No Limit For Protocol Values	#0002	Medium Risk	Closed
Unbounded Loop	#0003	Low Risk	Open
No Events Emitted For Changes To Protocol Values	#0004	Informational	Closed
SafeMath Unnecessary In Solidity 0.8	#0005	Informational	Open
Pairs Can Have Different Router Addresses	#0006	Low Risk	Closed

On-Chain Analysis

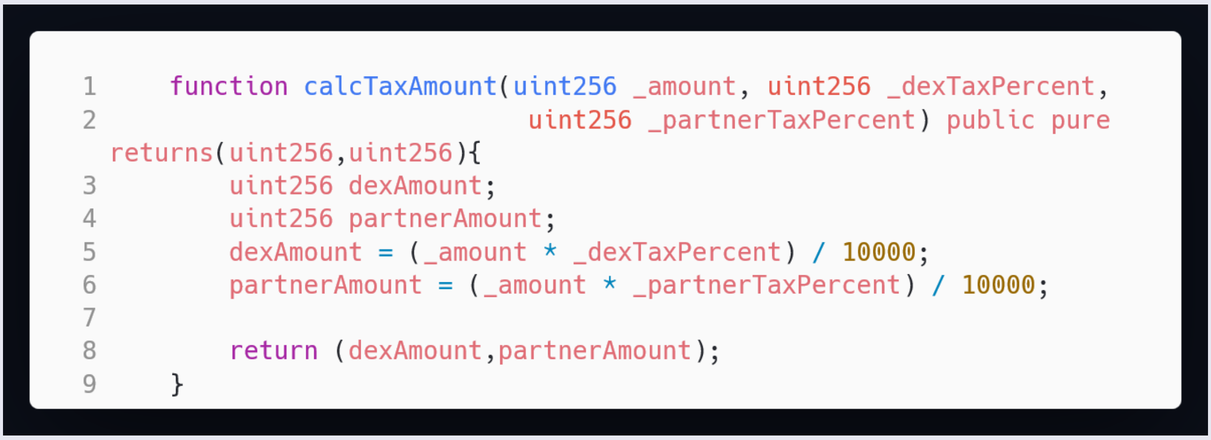
Finding	ID	Severity	Status
No Timelock	#0007	Medium Risk	Open
Unverified Contracts	#0008	Medium Risk	Open

Findings

Code Analysis

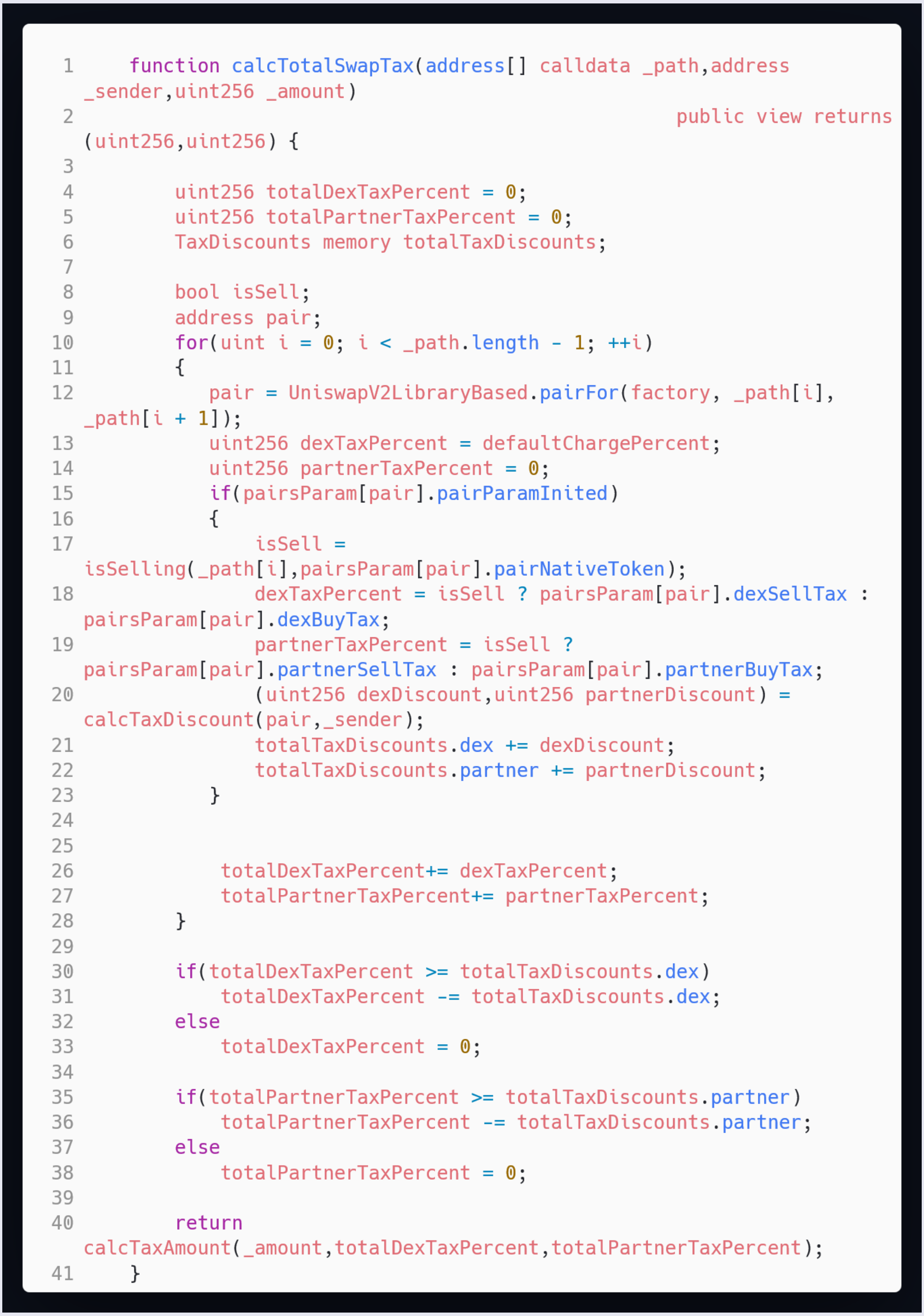
Taxes Can Be Used To Drain User Funds

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	UniswapV2Router.sol -> 303-311



```
    function calcTaxAmount(uint256 _amount, uint256 _dexTaxPercent,
                               uint256 _partnerTaxPercent) public pure
returns(uint256,uint256){
        uint256 dexAmount;
        uint256 partnerAmount;
        dexAmount = (_amount * _dexTaxPercent) / 10000;
        partnerAmount = (_amount * _partnerTaxPercent) / 10000;

        return (dexAmount,partnerAmount);
    }
```

LOCATION	UniswapV2Router.sol -> 349-389

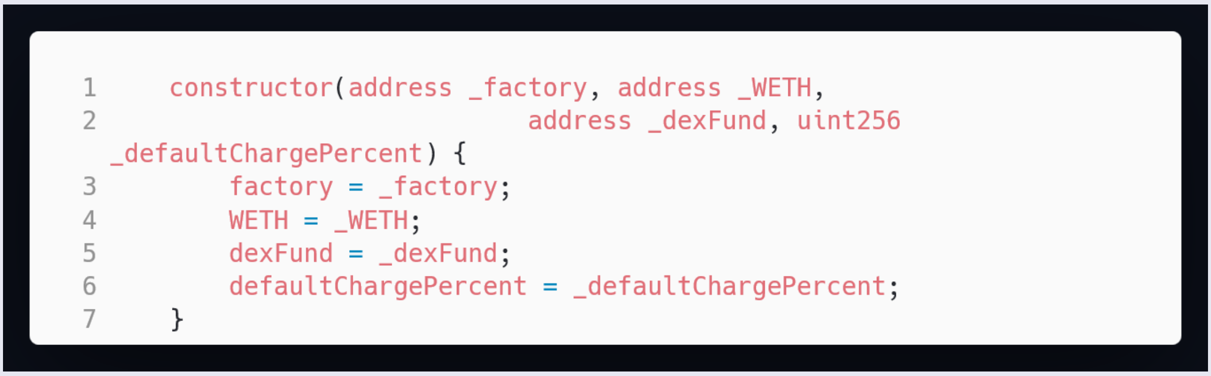


```
    function calcTotalSwapTax(address[] calldata _path,address _sender,uint256 _amount)
                                                    public view returns (uint256,uint256) {

        uint256 totalDexTaxPercent = 0;
        uint256 totalPartnerTaxPercent = 0;
        TaxDiscounts memory totalTaxDiscounts;

        bool isSell;
        address pair;
        for(uint i = 0; i < _path.length - 1; ++i)
        {
           pair = UniswapV2LibraryBased.pairFor(factory, _path[i], _path[i + 1]);
           uint256 dexTaxPercent = defaultChargePercent;
           uint256 partnerTaxPercent = 0;
           if(pairsParam[pair].pairParamInited)
           {
               isSell = isSelling(_path[i],pairsParam[pair].pairNativeToken);
               dexTaxPercent = isSell ? pairsParam[pair].dexSellTax : pairsParam[pair].dexBuyTax;
               partnerTaxPercent = isSell ? pairsParam[pair].partnerSellTax : pairsParam[pair].partnerBuyTax;
               (uint256 dexDiscount,uint256 partnerDiscount) = calcTaxDiscount(pair,_sender);
               totalTaxDiscounts.dex += dexDiscount;
               totalTaxDiscounts.partner += partnerDiscount;
           }


            totalDexTaxPercent+= dexTaxPercent;
            totalPartnerTaxPercent+= partnerTaxPercent;
        }

        if(totalDexTaxPercent >= totalTaxDiscounts.dex)
            totalDexTaxPercent -= totalTaxDiscounts.dex;
        else
            totalDexTaxPercent = 0;

        if(totalPartnerTaxPercent >= totalTaxDiscounts.partner)
            totalPartnerTaxPercent -= totalTaxDiscounts.partner;
        else
            totalPartnerTaxPercent = 0;

        return calcTaxAmount(_amount,totalDexTaxPercent,totalPartnerTaxPercent);
    }
```

DESCRIPTION	The total tax could be set arbitrarily high because it's the sum of all applicable taxes. Even if each one of the taxes has an upper bound, the total tax can be over 100% which could potentially lead to a drain of all user funds, no matter the swap amount (if the approval is high enough).
RECOMMENDATION	Set an upper bound on the final taxes.
RESOLUTION	The project team added an upper bound to the total tax of 90%.

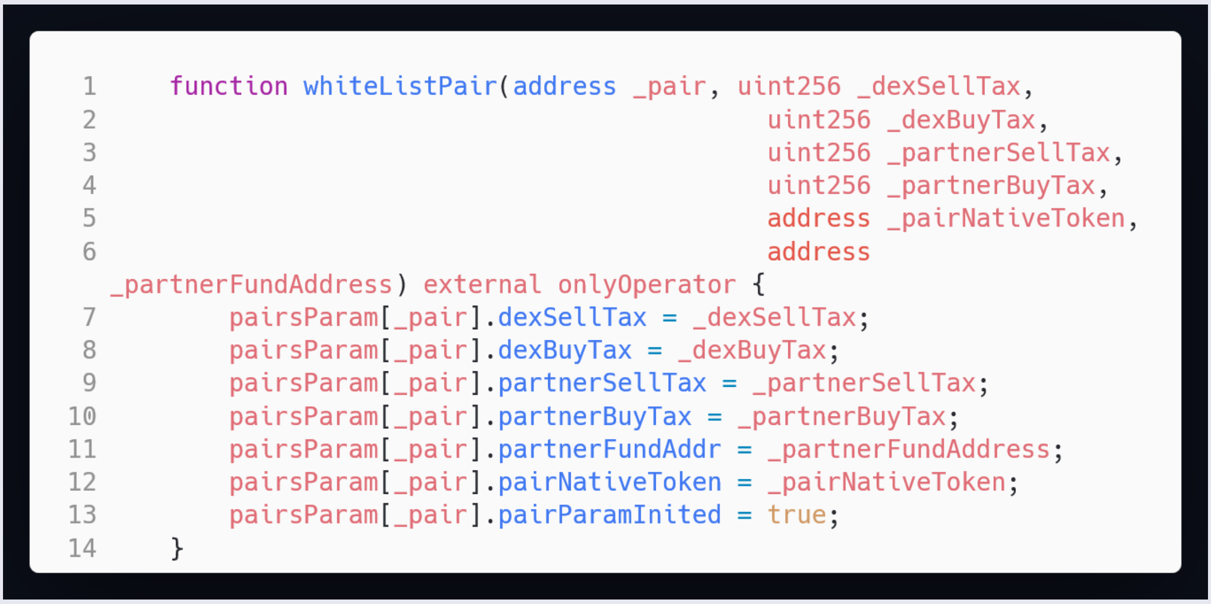
No Limit For Protocol Values

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	UniswapV2Router.sol -> 56-62



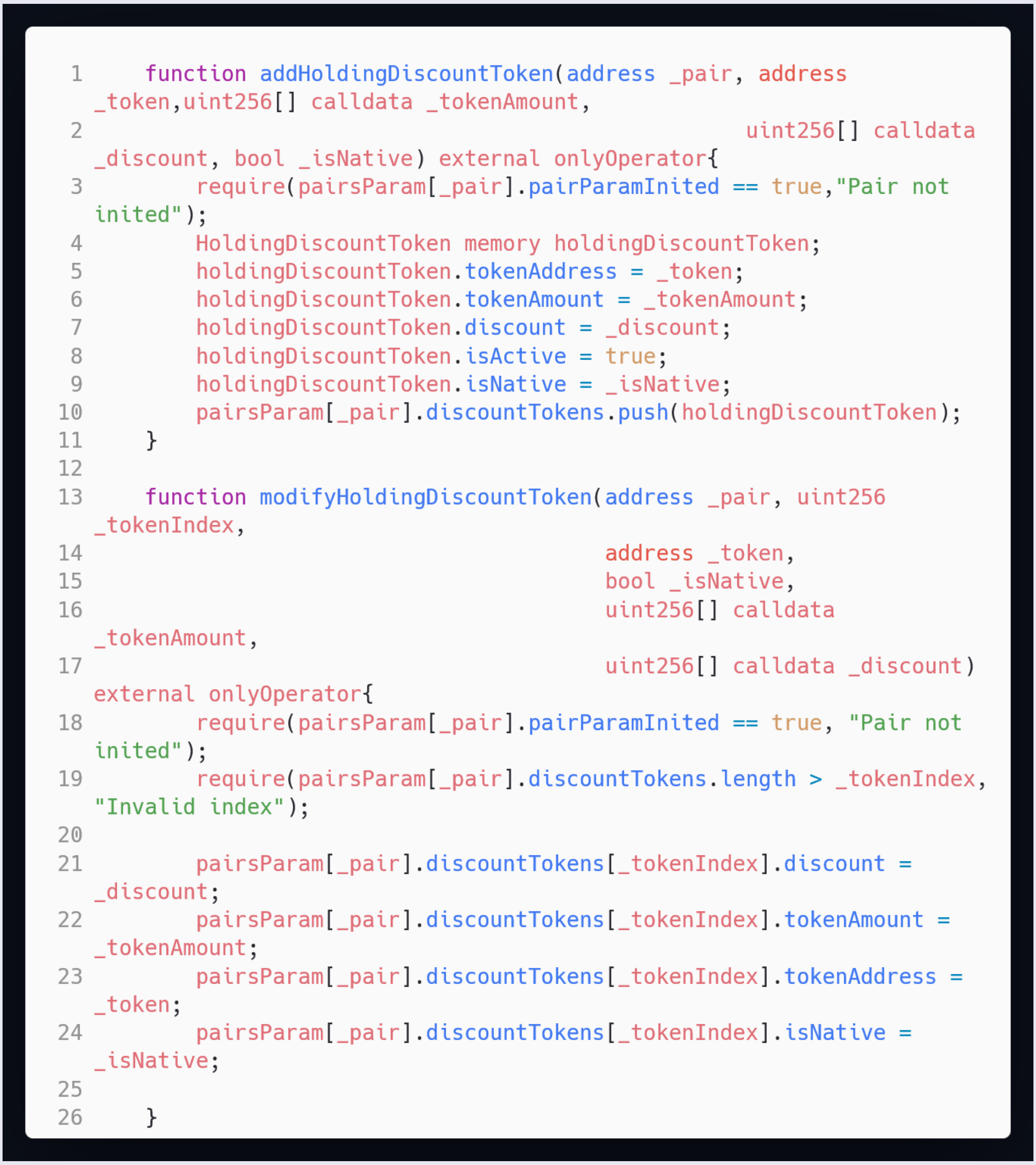
```
    constructor(address _factory, address _WETH,
                         address _dexFund, uint256 _defaultChargePercent) {
        factory = _factory;
        WETH = _WETH;
        dexFund = _dexFund;
        defaultChargePercent = _defaultChargePercent;
    }
```

LOCATION	UniswapV2Router.sol -> 246-259



```
    function whiteListPair(address _pair, uint256 _dexSellTax,
                                        uint256 _dexBuyTax,
                                        uint256 _partnerSellTax,
                                        uint256 _partnerBuyTax,
                                        address _pairNativeToken,
                                        address _partnerFundAddress) external onlyOperator {
        pairsParam[_pair].dexSellTax = _dexSellTax;
        pairsParam[_pair].dexBuyTax = _dexBuyTax;
        pairsParam[_pair].partnerSellTax = _partnerSellTax;
        pairsParam[_pair].partnerBuyTax = _partnerBuyTax;
        pairsParam[_pair].partnerFundAddr = _partnerFundAddress;
        pairsParam[_pair].pairNativeToken = _pairNativeToken;
        pairsParam[_pair].pairParamInited = true;
    }
```

LOCATION	UniswapV2Router.sol -> 266-291



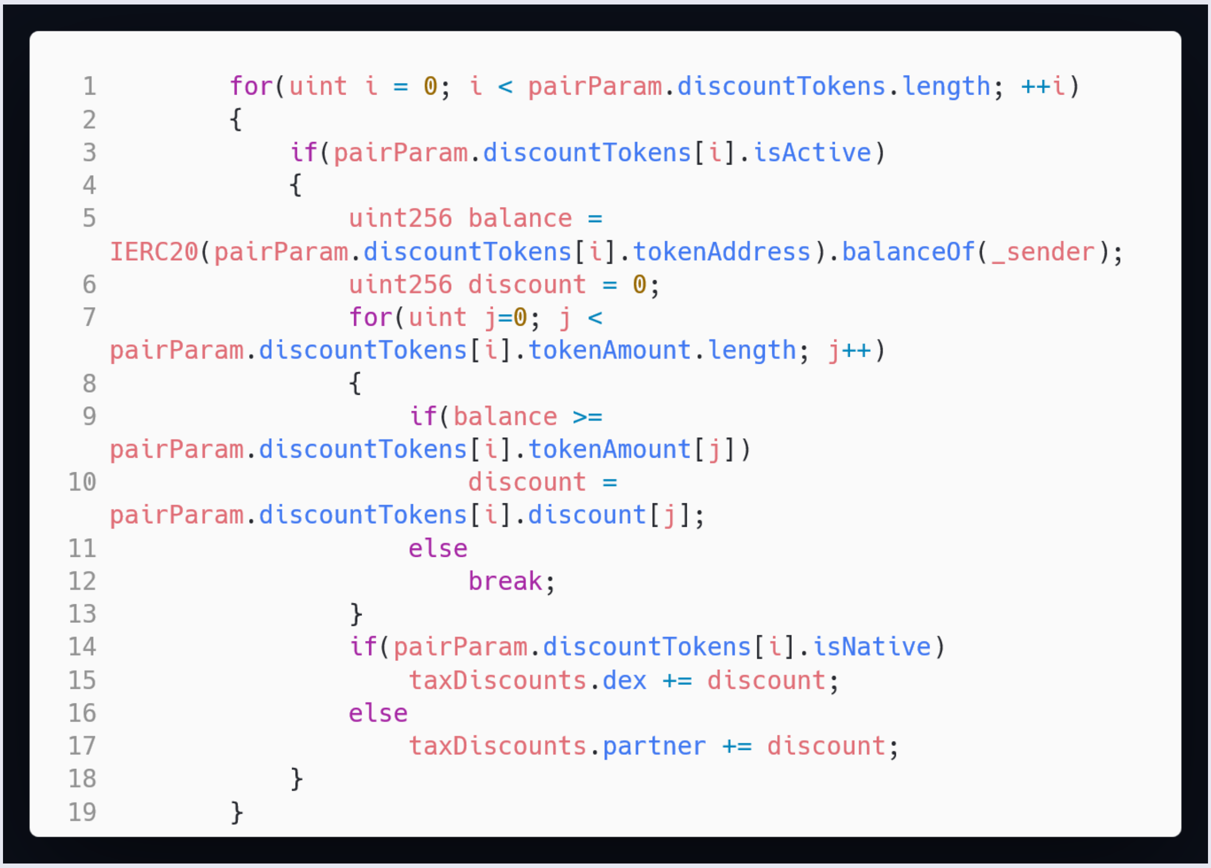
```solidity
    function addHoldingDiscountToken(address _pair, address _token,uint256[] calldata _tokenAmount,
                                                        uint256[] calldata _discount, bool _isNative) external onlyOperator{
        require(pairsParam[_pair].pairParamInited == true,"Pair not inited");
        HoldingDiscountToken memory holdingDiscountToken;
        holdingDiscountToken.tokenAddress = _token;
        holdingDiscountToken.tokenAmount = _tokenAmount;
        holdingDiscountToken.discount = _discount;
        holdingDiscountToken.isActive = true;
        holdingDiscountToken.isNative = _isNative;
        pairsParam[_pair].discountTokens.push(holdingDiscountToken);
    }

    function modifyHoldingDiscountToken(address _pair, uint256 _tokenIndex,
                                        address _token,
                                        bool _isNative,
                                        uint256[] calldata _tokenAmount,
                                        uint256[] calldata _discount) external onlyOperator{
        require(pairsParam[_pair].pairParamInited == true, "Pair not inited");
        require(pairsParam[_pair].discountTokens.length > _tokenIndex, "Invalid index");

        pairsParam[_pair].discountTokens[_tokenIndex].discount = _discount;
        pairsParam[_pair].discountTokens[_tokenIndex].tokenAmount = _tokenAmount;
        pairsParam[_pair].discountTokens[_tokenIndex].tokenAddress = _token;
        pairsParam[_pair].discountTokens[_tokenIndex].isNative = _isNative;

    }
```

DESCRIPTION	Parameters that are percentages have no upper bound. Arbitrarily high values can break the protocol functionality of the contract.
RECOMMENDATION	Add an upper bound requires statement for *defaultChargePercent*, *pairsParam[_pair].dexSellTax*, *pairsParam[_pair].dexBuyTax*, *pairsParam[_pair].partnerSellTax*, *pairsParam[_pair].partnerBuyTax*,

	holdingDiscountToken.discount and *pairsParam[_pair].discountTokens[_tokenIndex].discount*.
RESOLUTION	The project team has implemented the recommended fix.

Unbounded Loop

FINDING ID	#0003
SEVERITY	Low Risk
STATUS	Open
LOCATION	UniswapV2Router.sol -> 317-335



```
        for(uint i = 0; i < pairParam.discountTokens.length; ++i)
        {
            if(pairParam.discountTokens[i].isActive)
            {
                uint256 balance = IERC20(pairParam.discountTokens[i].tokenAddress).balanceOf(_sender);
                uint256 discount = 0;
                for(uint j=0; j < pairParam.discountTokens[i].tokenAmount.length; j++)
                {
                    if(balance >= pairParam.discountTokens[i].tokenAmount[j])
                        discount = pairParam.discountTokens[i].discount[j];
                    else
                        break;
                }
                if(pairParam.discountTokens[i].isNative)
                    taxDiscounts.dex += discount;
                else
                    taxDiscounts.partner += discount;
            }
        }
```

DESCRIPTION	Iterating over an unbounded array can cause transactions to revert due to the gas limit.
RECOMMENDATION	It is recommended to add an input variable to the functions that bound the loop to a max limit.
RESOLUTION	The project team did not solve this issue.

No Events Emitted For Changes To Protocol Values

FINDING ID	#0004
SEVERITY	Informational
STATUS	Closed
LOCATION	● UniswapV2Factory.sol -> 24: *function setRouter(address _router) external onlyOwner {* ● UniswapV2Factory.sol -> 28: *function updatePairsRouter() external onlyOwner{* ● UniswapV2Factory.sol -> 35: *function updatePairRouter(uint index) external onlyOwner{* ● UniswapV2Pair.sol -> 164: *function setRouter(address _router) external onlyOwner {* ● UniswapV2Router.sol -> 246: *function whiteListPair(address _pair, uint256 _dexSellTax, uint256 _dexBuyTax, uint256 _partnerSellTax, uint256 _partnerBuyTax, address _pairNativeToken, address _partnerFundAddress) external onlyOperator {* ● UniswapV2Router.sol -> 261: *function deListPair(address _pair) external onlyOperator {* ● UniswapV2Router.sol -> 266: *function addHoldingDiscountToken(address _pair, address _token,uint256[] calldata _tokenAmount, uint256[] calldata _discount, bool _isNative) external onlyOperator{* ● UniswapV2Router.sol -> 278: *function modifyHoldingDiscountToken(address _pair, uint256 _tokenIndex, address _token, bool _isNative, uint256[] calldata _tokenAmount, uint256[] calldata _discount) external onlyOperator{* ● UniswapV2Router.sol -> 293: *function toggleHoldingDiscountToken(address _pair,uint256 _tokenIndex, bool _isActive) external onlyOperator{*

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.

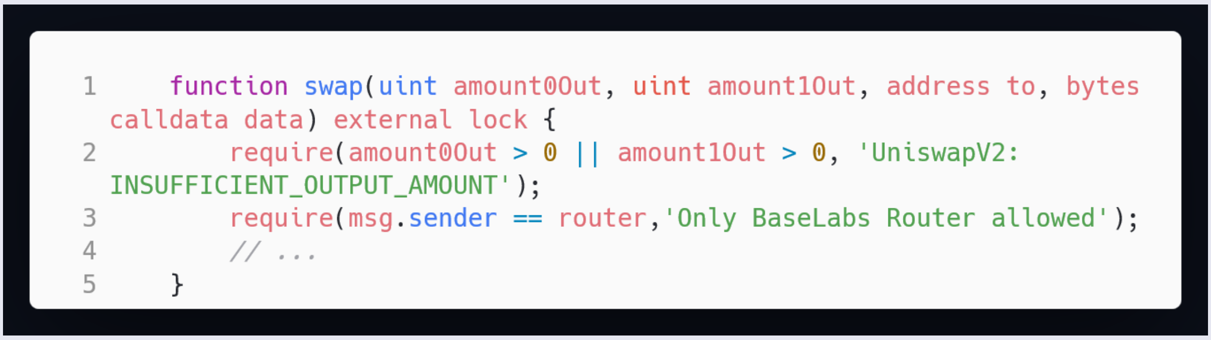
SafeMath Unnecessary In Solidity 0.8

FINDING ID	#0005
SEVERITY	Informational
STATUS	Open
LOCATION	● UniswapV2Router.sol -> 39: *using SafeMath for uint;* ● UniswapV2LibraryBased.sol -> 9: *using SafeMath for uint256;* ● UniswapV2Pair.sol -> 16: *using SafeMath for uint;*

DESCRIPTION	SafeMath is no longer necessary in solidity 0.8.
RECOMMENDATION	Remove SafeMath.
RESOLUTION	The project team did not solve this issue.

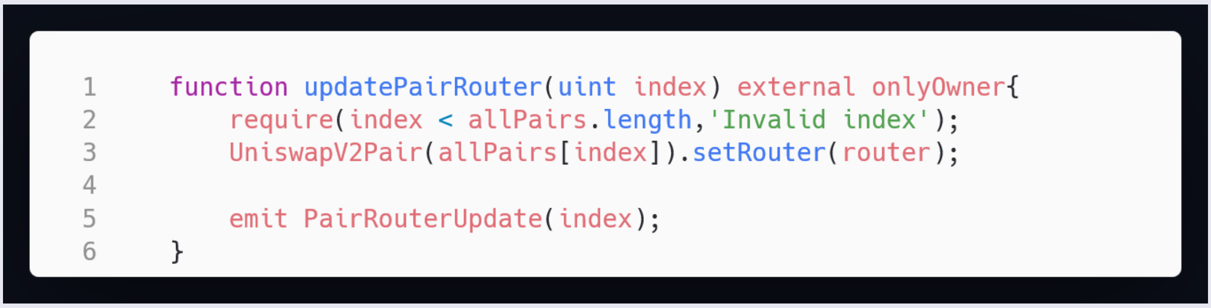
Pairs Can Have Different Router Addresses

FINDING ID	#0006
SEVERITY	Low Risk
STATUS	Closed
LOCATION	UniswapV2Pair.sol -> 173-203



```
    function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data) external lock {
        require(amount0Out > 0 || amount1Out > 0, 'UniswapV2: INSUFFICIENT_OUTPUT_AMOUNT');
        require(msg.sender == router,'Only BaseLabs Router allowed');
        // ...
    }
```

LOCATION	UniswapV2Factory.sol -> 41-46



```
    function updatePairRouter(uint index) external onlyOwner{
        require(index < allPairs.length,'Invalid index');
        UniswapV2Pair(allPairs[index]).setRouter(router);

        emit PairRouterUpdate(index);
    }
```

DESCRIPTION	The factory is responsible for setting and updating the router address of the pairs. Updating the router address in the factory will cause all subsequent pairs to use the new router, but the previous pairs will still have the old router. Only the pairs assigned router can swap on it, and therefore swapping on a path with multiple assigned routers will revert.
RECOMMENDATION	Ensure that all pairs have the same router. (Note that the factory's *updatePairsRouter()* function is liable to revert due to the gas limit). Alternatively, make all pairs retrieve the router address from the factory.
RESOLUTION	The team is now retrieving the router address from the

	factory contract. The *UniswapV2Pair.sol* still has the router variable, its setter method and the *initialize()* function initializes its value.

On-Chain Analysis

No Timelock

FINDING ID	#0007
SEVERITY	Medium Risk
STATUS	Open
LOCATION	UniswapV2Factory 0x407C47E3FDB7952Ee53aa232B5f28566A024A759 UniswapV2Router 0xaf2098E3AF3AC27a3a9362ACa2D883eFC36c7FAC

DESCRIPTION	The noted contracts have not had their ownership transferred to a timelock contract. The router has a number of settings including tax rates, discount rates, and whitelisted tokens which can be changed without any delay or notice. Tax rates could be changed to front-run a high-value swap. The function that owner/feeToSetter of UniswapV2Factory.sol can call is: ● *setRouter(address _router)* The functions that owner/operator of UniswapV2Router.sol can call are: ● *whiteListPair(address _pair, uint256 _dexSellTax, uint256 _dexBuyTax, uint256 _partnerSellTax, uint256 _partnerBuyTax, address _pairNativeToken, address _partnerFundAddress)* ● *deListPair(address _pair)* ● *addHoldingDiscountToken(address _pair, address _token,uint256[] calldata _tokenAmount, uint256[] calldata _discount, bool _isNative)* ● *modifyHoldingDiscountToken(address _pair, uint256 _tokenIndex, address _token, bool _isNative, uint256[] calldata _tokenAmount, uint256[] calldata _discount)* ● *toggleHoldingDiscountToken(address _pair,uint256 _tokenIndex, bool _isActive)* Owner address: 0xB9B22504b9071291E938E0E582934A82C4a4670c

RECOMMENDATION	Transfer ownership to a timelock contract. Obelisk recommends a timelock delay for all functionality of at least 72 hours.
RESOLUTION	N/A

Unverified Contracts

FINDING ID	#0008
SEVERITY	Medium Risk
STATUS	Open
LOCATION	UniswapV2Factory 0x407C47E3FDB7952Ee53aa232B5f28566A024A759 UniswapV2Router 0xaf2098E3AF3AC27a3a9362ACa2D883eFC36c7FAC

DESCRIPTION	Noted contracts were not verified on the explorer. Obelisk verified that these contracts match the contracts of the audit.
RECOMMENDATION	Verify these contracts or make the contract repository public.
RESOLUTION	N/A

External Addresses

Externally Owned Accounts

Owner

ACCOUNT	0xB9B22504b9071291E938E0E582934A82C4a4670c
USAGE	0x407C47E3FDB7952Ee53aa232B5f28566A024A759 *UniswapV2Factory.feeToSetter* *UniswapV2Factory.owner* 0xaf2098E3AF3AC27a3a9362ACa2D883eFC36c7FAC *UniswapV2Router.operator* *UniswapV2Router.owner*
IMPACT	● receives elevated permissions as owner, operator, or other

ACCOUNT	0xc5bf5A56cEc938312A2528e2Bf5A5CBcaBEb33ea
USAGE	0x407C47E3FDB7952Ee53aa232B5f28566A024A759 *UniswapV2Factory.feeTo* 0xaf2098E3AF3AC27a3a9362ACa2D883eFC36c7FAC *UniswapV2Router.dexFund*
IMPACT	● receives elevated permissions as owner, operator, or other

External Contracts

These contracts are not part of the audit scope.

No External Contracts

External Tokens

These contracts are not part of the audit scope.

Wrapped FTM

ADDRESS	WFTM - 0x21be370d5312f44cb42ce377bc9b8a0cef1a4c83
USAGE	0xaf2098E3AF3AC27a3a9362ACa2D883eFC36c7FAC *UniswapV2Router.WETH*
IMPACT	● ERC20 Token

Appendix A - Reviewed Documents

Deployed Contracts

Document	Address
dex/UniswapV2Factory.sol	0x407C47E3FDB7952Ee53aa232B5f28566A024A759
dex/UniswapV2Router.sol	0xaf2098E3AF3AC27a3a9362ACa2D883eFC36c7FAC

Libraries And Interfaces

dex/UniswapV2ERC20.sol dex/UniswapV2Pair.sol interfaces/IERC20.sol interfaces/IUniswapV2Callee.sol interfaces/IUniswapV2ERC20.sol interfaces/IUniswapV2Factory.sol interfaces/IUniswapV2Pair.sol interfaces/IUniswapV2Router.sol interfaces/IWeth.sol lib/Math.sol lib/UniswapV2LibraryBased.sol lib/UQ112x112.sol owner/Operator.sol

Revisions

Revision 1	6efeb9e9d9c3392439e1693f95324a641392907f
Revision 2	cc5e6a8d5fafef94ee057845529856c0e423b541
Revision 3	ab13a15835b2f0f4e33a0c2729d42a86622ea404
Revision 4	ac836cacc16eec5f3d021654fecee2637fb108c3
Revision 5	a6ba179229d2b359de96e5ce81555866f481dc3d
Revision 6	113b9a966d36c798bd133e093b9dbb7e348af0ac

Imported Contracts

openzeppelin	OpenZeppelin: 4.5.0
uniswap/v2-core	Uniswap/lib: 4.0.1-alpha

uniswap/lib	Uniswap/v2-core: 1.0.1

Appendix B - Risk Ratings

Risk	Description
High Risk	Security risks that are ***almost certain*** to lead to ***impairment or loss of funds***. Projects are advised to fix as soon as possible.
Medium Risk	Security risks that are ***very likely*** to lead to ***impairment or loss of funds*** with ***limited impact***. Projects are advised to fix as soon as possible.
Low Risk	Security risks that can lead to ***damage to the protocol***. Projects are advised to fix. Issues with this rating might be used in an exploit with other issues to cause significant damage.
Informational	Noteworthy information. Issues may include code conventions, missing or conflicting information, gas optimizations, and other advisories.

Appendix C - Finding Statuses

Closed	Contracts were modified to permanently resolve the finding.
Mitigated	The finding was resolved on-chain. The issue may require monitoring, for example in the case of a time lock.
Partially Closed	Contracts were modified to partially fix the issue
Partially Mitigated	The finding was resolved by project specific methods which cannot be verified on chain. Examples include compounding at a given frequency, or the use of a multisig wallet.
Open	The finding was not addressed.

Appendix D - Glossary

Contract Structure

Contract: An address with which provides functionality to users and other contracts. They are implemented in code and deployed to the blockchain.
Protocol: A system of contracts which work together.
Stakeholders: The users, operators, owners, and other participants of a contract.

Security Concepts

Bug**:** A defect in the contract code.
Exploit**:** A chain of events involving bugs, vulnerabilities, or other security risks which damages a protocol.
Funds: Tokens deposited by users or other stakeholders into a protocol.
Impairment: The loss of functionality in a contract or protocol.
Security risk: A circumstance that may result in harm to the stakeholders of a protocol. Examples include vulnerabilities in the code, bugs, excessive permissions, missing timelock, etc.
Vulnerability: A vulnerability is a flaw that allows an attacker to potentially cause harm to the stakeholders of a contract. They may occur in a contract's code, design, or deployed state on the blockchain.

Appendix E - 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