OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 61	2022-03-30	ByFixter, Donut	Audit Final

Audit Notes

Audit Date	2022-03-10 - 2022-03-29
Auditor/Auditors	ByFixter, 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	OB12457852

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	Fantasm
Description	Fantastic Protocol is a DeFi project aimed at developing and popularizing synthetic tokens for multiple L1 Ecosystems. Imagine having exposure to the price of FTM or AVAX token without actually owning it.
Website	https://fantasm.finance/
Contact	@fantasm_finance on Twitter
Contact information	@FantasmCat_NOdm on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Fantom

Audit of Fantasm

Obelisk was commissioned by Fantasm on the 5th of March 2022 to conduct a comprehensive audit of Fantasm' contracts. The following audit was conducted between the 10th of March 2022 and the 29th 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.

Many of the findings have been solved by the project team by implementing fixes within the code or deploying changes. There are still some risks of varying severity left open or partially open. These issues do have comments attached to them that the reader is encouraged to read through to understand the underlying risk.

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 Fantasm 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

Manual Analysis

Finding	ID	Severity	Status
Minting And Redeeming Uses Spot Price	#0001	High Risk	Partially Closed
Minting And Redeeming Can Be Disabled Arbitrarily	#0002	High Risk	Open
Oracle Can Be Invalid When Re-collateralizing	#0003	High Risk	Open
Oracle Can Return Stale Price	#0004	Medium Risk	Closed
Uniswap Oracle Requires Overflow	#0005	Medium Risk	Closed
No Limit For Protocol Values	#0006	Medium Risk	Closed
Treasury Strategy Addresses Cannot Easily Be Identified	#0007	Medium Risk	Closed
Swapping Tokens Can Be Frontrun	#0008	Medium Risk	Closed
No emergencyWithdraw Function	#0009	Medium Risk	Open
Minters Addresses Cannot Easily Be Identified	#0010	Low Risk	Partially Mitigated
Setters Could Cause Deployment Issues	#0011	Low Risk	Partially Closed
Use Safe Transfer for native currency	#0012	Low Risk	Partially Closed
Spot Price Normalized To 18 Decimals	#0013	Low Risk	Partially Closed
Unconventional Burn Method	#0014	Low Risk	Closed
Unbounded Loop	#0015	Low Risk	Partially Closed
Comment Does Not Match Ether Allocation	#0016	Informational	Mitigated

rewardDistributors Addresses Cannot Easily Be Identified	#0017	Informational	Open
Swaps always execute on the same block	#0018	Informational	Closed
SafeMath Unnecessary In Solidity 0.8	#0019	Informational	Partially Closed
Function Should Be Marked As View	#0020	Informational	Closed
Unused Library	#0021	Informational	Closed
No Events Emitted For Changes To Protocol Values	#0022	Informational	Closed
Unused Event	#0023	Informational	Closed
Initial FSM Distribution	#0024	High Risk	Mitigated
SwapStrategyPOL Slippage Calculation	#0025	Medium Risk	Closed
Use safeTransfer for ERC20 transfer	#0026	Medium Risk	Closed
Whitelisted Addresses Can Bypass Flashloan Protection	#0027	High Risk	Closed
Timelock Delay Is Short	#0028	High Risk	Open
Timelock Minimum Delay Is Short	#0029	Medium Risk	Open
FXM Treasury Fund Owner is an Unverified Contract	#0030	Medium Risk	Open
Initial Slippage is Out Of Bounds	#0031	Medium Risk	Mitigated
TWAP Price Normalized To 18 Decimals	#0032	Low Risk	Mitigated
Minor Changes To Contracts	#0033	Informational	Closed

Static Analysis

Finding	ID	Severity	Status
Function Should Be Marked As	#0020	Informational	Closed

View			
Unused Library	#0021	Informational	Closed
No Events Emitted For Changes To Protocol Values	#0022	Informational	Closed
Unused Event	#0023	Informational	Closed
Initial FSM Distribution	#0024	High Risk	Mitigated
SwapStrategyPOL Slippage Calculation	#0025	Medium Risk	Closed
Use safeTransfer for ERC20 transfer	#0026	Medium Risk	Closed
Whitelisted Addresses Can Bypass Flashloan Protection	#0027	High Risk	Closed

On-Chain Analysis

Finding	ID	Severity	Status
Timelock Delay Is Short	#0028	High Risk	Open
Timelock Minimum Delay Is Short	#0029	Medium Risk	Open
FXM Treasury Fund Owner is an Unverified Contract	#0030	Medium Risk	Open
Inital Slippage is Out Of Bounds	#0031	Medium Risk	Mitigated
TWAP Price Normalized To 18 Decimals	#0032	Low Risk	Mitigated
Minor Changes To Contracts	#0033	Informational	Closed

Findings

Manual Analysis

Minting And Redeeming Uses Spot Price

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Partially Closed
LOCATION	oracles/UniswapPairOracle.sol -> 70-86

```
1     function spot(address token, uint256 pricePrecision) external view
  returns (uint256 amountOut) {
2         IUniswapV2Pair uniswapPair = IUniswapV2Pair(pair);
3         address _token0 = uniswapPair.token0();
4         address _token1 = uniswapPair.token1();
5         require(_token0 == token || _token1 == token, "Invalid pair");
6         (uint256 _reserve0, uint256 _reserve1, ) =
  uniswapPair.getReserves();
7         require(_reserve0 > 0 && _reserve1 > 0, "No reserves");
8         uint8 _token0MissingDecimals = 18 -
  (ERC20(_token0).decimals());
9         uint8 _token1MissingDecimals = 18 -
  (ERC20(_token1).decimals());
10        uint256 _price = 0;
11        if (token == _token0) {
12            _price =
  _reserve1.mul(10**_token1MissingDecimals).mul(pricePrecision).div(_rese
  rve0);
13        } else {
14            _price =
  _reserve0.mul(10**_token0MissingDecimals).mul(pricePrecision).div(_rese
  rve1);
15        }
16        return _price;
17    }
```

LOCATION	oracles/MasterOracle.sol -> 33-39

```
    function getXTokenPrice() public view returns (uint256) {
        return oracleXToken.spot(xToken, PRICE_PRECISION);
    }

    function getYTokenPrice() public view returns (uint256) {
        return oracleYToken.spot(yToken, PRICE_PRECISION);
    }
```

LOCATION	Pool.sol -> 136 Pool.sol -> 160

```
        uint256 _yTokenPrice = oracle.getYTokenPrice();
```

DESCRIPTION	The minting and redeeming calculate ratios using the spot price. This price is vulnerable to manipulation including flashloans. An attacker may manipulate the price in order to mint more tokens than expected, which may ultimately result in draining the reserves. While there is an anti-flashloan mechanism, it only protects against collecting rewards within the same block as minting or redeeming. It will not protect against manipulations that only deal with minting and redeeming.
RECOMMENDATION	Use the TWAP price for minting and redeeming. Note that the TWAP should be updated frequently in order to prevent arbitrage between the TWAP and the spot price.
RESOLUTION	The oracle can no longer generate prices if its update is older than 12 hours. Furthermore, price drifts of over 10% between the spot and TWAP disables redemption. This should provide resistance against extreme price manipulation, but does not provide full protection.

Minting And Redeeming Can Be Disabled Arbitrarily

FINDING ID	#0002
SEVERITY	High Risk
STATUS	Open
LOCATION	Pool.sol -> 223-245

```
1     function mint(uint256 _minXTokenOut) external payable nonReentrant
  {
2         require(!mintPaused, "Pool::mint: Minting is paused");
3         // ...
4     }
```

LOCATION	Pool.sol -> 247-278

```
1     function redeem(
2         uint256 _xTokenIn,
3         uint256 _minYTokenOut,
4         uint256 _minEthOut
5     ) external nonReentrant {
6         require(!redeemPaused, "Pool::redeem: Redeeming is paused");
7         // ...
8     }
```

LOCATION	Pool.sol -> 343-347

```
1     function toggle(bool _mintPaused, bool _redeemPaused) public
  onlyOwner {
2         mintPaused = _mintPaused;
3         redeemPaused = _redeemPaused;
4         emit Toggled(_mintPaused, _redeemPaused);
5     }
```

DESCRIPTION	The mint and redeem functions of the Pool contract can be disabled by the contract owner. Disabling without warning may prevent users from withdrawing their deposits.
RECOMMENDATION	The following options can be used to resolve this issue. • Ensure that a proper timelock mechanism is set as the contract owner. Obelisk recommends a 72-hour timelock.

	• Prevent these functions from being disabled once enabled. • Add a maximum amount of time the contract can be paused. Remember to add a grace period where the contract must be unlocked.
RESOLUTION	Project team: “It can be either good or bad. Protocol can arbitrary paused the functioning of the contracts for either good or bad purposes.”

Oracle Can Be Invalid When Recollateralizing

FINDING ID	#0003
SEVERITY	High Risk
STATUS	Open
LOCATION	Pool.sol -> 190-216

```
1     function refreshCollateralRatio() public {
2         require(collateralRatioPaused == false,
  "Pool::refreshCollateralRatio: Collateral Ratio has been paused");
3         require(block.timestamp - lastRefreshCrTimestamp >=
  refreshCooldown, "Pool::refreshCollateralRatio: Must wait for the
  refresh cooldown since last refresh");
4
5         uint256 _xTokenPrice = oracle.getXTokenTWAP();
6         if (_xTokenPrice > priceTarget + priceBand) {
7             if (collateralRatio <= ratioStepDown) {
8                 collateralRatio = 0;
9             } else {
10                uint256 _newCR = collateralRatio - ratioStepDown;
11                if (_newCR <= minCollateralRatio) {
12                    collateralRatio = minCollateralRatio;
13                } else {
14                    collateralRatio = _newCR;
15                }
16            }
17        } else if (_xTokenPrice < priceTarget - priceBand) {
18            if (collateralRatio + ratioStepUp >= COLLATERAL_RATIO_MAX)
  {
19                collateralRatio = COLLATERAL_RATIO_MAX;
20            } else {
21                collateralRatio = collateralRatio + ratioStepUp;
22            }
23        }
24
25        lastRefreshCrTimestamp = block.timestamp;
26        emit NewCollateralRatioSet(collateralRatio);
27    }
```

LOCATION	Pool.sol -> 415-419

```
1     function setOracle(IMasterOracle _oracle) external onlyOwner {
2         require(address(_oracle) != address(0), "Pool::setOracle:
  invalid address");
3         oracle = _oracle;
4         emit OracleChanged(address(_oracle));
5     }
```

LOCATION	oracles/UniswapPairOracle.sol -> 44-58

```
1     function update() external {
2         (uint256 price0Cumulative, uint256 price1Cumulative, uint32
  blockTimestamp) = currentCumulativePrices(address(pair));
3         uint32 timeElapsed = blockTimestamp - blockTimestampLast; //
  Overflow is desired
4
5         // Ensure that at least one full period has passed since the
  last update
6         require(timeElapsed >= PERIOD, "PairOracle:
  PERIOD_NOT_ELAPSED");
7
8         // Overflow is desired, casting never truncates
9         // Cumulative price is in (uq112x112 price * seconds) units so
  we simply wrap it after division by time elapsed
10        price0Average = FixedPoint.uq112x112(uint224((price0Cumulative
  - price0CumulativeLast) / timeElapsed));
11        price1Average = FixedPoint.uq112x112(uint224((price1Cumulative
  - price1CumulativeLast) / timeElapsed));
12        price0CumulativeLast = price0Cumulative;
13        price1CumulativeLast = price1Cumulative;
14        blockTimestampLast = blockTimestamp;
15    }
```

DESCRIPTION	The oracle can be switched out during the operation of the contract. However, the UniswapPairOracle requires that the oracle has updated at least once before it is able to produce a time-weighted price.
RECOMMENDATION	Ensure that the oracle has been updated before attaching a new oracle to the Pool. Alternatively, disable changing the Pool's oracle.
RESOLUTION	Project Team: “By using oracles with prices were not updated ($0), the minting / redemption will be reverted, the pool functions will be protected.“ Obelisk Team: “The minting and redeeming will be blocked, but the collateral ratio can still be updated in the mean time.”

Oracle Can Return Stale Price

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	oracles/UniswapPairOracle.sol -> 61-68

```
1     function twap(address token, uint256 pricePrecision) external view
  returns (uint256 amountOut) {
2         if (token == token0) {
3             amountOut = price0Average.mul(pricePrecision).decode144();
4         } else {
5             require(token == token1, "PairOracle: INVALID_TOKEN");
6             amountOut = price1Average.mul(pricePrecision).decode144();
7         }
8     }
```

DESCRIPTION	The *twap* method will return a "stale" (outdated) price if the last updated was significantly more than *PERIOD* seconds in the past.
RECOMMENDATION	Add a "leniency" period to the oracle; if the last update was more than *PERIOD+LENIENCY* seconds ago, revert on call to *twap*.
RESOLUTION	The oracle will revert when requested for a twap that is older than 12 hours.

Uniswap Oracle Requires Overflow

FINDING ID	#0005
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	oracles/UniswapPairOracle.sol -> 44-58

```
    function update() external {
        (uint256 price0Cumulative, uint256 price1Cumulative, uint32
blockTimestamp) = currentCumulativePrices(address(pair));
        uint32 timeElapsed = blockTimestamp - blockTimestampLast; //
Overflow is desired

        // Ensure that at least one full period has passed since the
last update
        require(timeElapsed >= PERIOD, "PairOracle:
PERIOD_NOT_ELAPSED");

        // Overflow is desired, casting never truncates
        // Cumulative price is in (uq112x112 price * seconds) units so
we simply wrap it after division by time elapsed
        price0Average = FixedPoint.uq112x112(uint224((price0Cumulative
- price0CumulativeLast) / timeElapsed));
        price1Average = FixedPoint.uq112x112(uint224((price1Cumulative
- price1CumulativeLast) / timeElapsed));
        price0CumulativeLast = price0Cumulative;
        price1CumulativeLast = price1Cumulative;
        blockTimestampLast = blockTimestamp;
    }
```

LOCATION	oracles/UniswapPairOracle.sol -> 109-117

```
        if (_blockTimestampLast != blockTimestamp) {
            // subtraction overflow is desired
            uint32 timeElapsed = blockTimestamp - _blockTimestampLast;
            // addition overflow is desired
            // counterfactual
            price0Cumulative += uint256(FixedPoint.fraction(reserve1,
reserve0)._x) * timeElapsed;
            // counterfactual
            price1Cumulative += uint256(FixedPoint.fraction(reserve0,
reserve1)._x) * timeElapsed;
        }
```

DESCRIPTION	A uniswap LP oracle is expected to make use of overflowing to deal with cumulative prices. However, from

	solidity 0.8.0, basic arithmetic operations automatically revert on overflow. If the update fails because of this, the oracle price will no longer update as expected and the treasury will operate with a stale price.
RECOMMENDATION	Add unchecked to the oracle logic where expected.
RESOLUTION	The project team has implemented the recommended fix.

No Limit For Protocol Values

FINDING ID	#0006
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	• farms/FantasticChef.sol -> 284-288: *function setRewardPerSecond(uint256 _rewardPerSecond) public onlyOwner {* • farms/FantasticChef.sol -> 247-260 : *function add(uint256 allocPoint, IERC20 _lpToken, IRewarder _rewarder) public onlyOwner* • farms/FantasticChef.sol -> 267-280: *function set(uint256 _pid, uint256 _allocPoint, IRewarder _rewarder, bool overwrite) public onlyOwner* • oracles/UniswapPairOracle.sol -> 40-42: *function setPeriod(uint256 _period) external onlyOwner {*

DESCRIPTION	The following values can be set arbitrarily high, potentially breaking the functionality of the contracts: Note for *UniswapPairOracle.PERIOD*: When using an on-chain single oracle two important factors are (Accuracy, Security). Increasing the time will raise the security but lower the accuracy. Decreasing time will lower security but raise accuracy. Note For *FantasticChef.add* and *FantasticChef.set*: Missing limit for *allocPoint*
RECOMMENDATION	Add bounds to the values.
RESOLUTION	Upper limits are added to *FantasticChef.setRewardPerSecond* Upper and lower limits are added to *UniswapPairOracle.PERIOD*

Treasury Strategy Addresses Cannot Easily Be Identified

FINDING ID	#0007
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	• FantasticTreasury.sol -> 19: *mapping(address => bool) public strategies;*

DESCRIPTION	The treasury strategy addresses noted are not easily identified by users. It is important to ensure that all addresses are accounted for.
RECOMMENDATION	Add an enumerated list of all treasury strategy addresses. Add events that emit when an address is added or removed.
RESOLUTION	The project team has implemented the recommended fix.

Swapping Tokens Can Be Frontrun

FINDING ID	#0008
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	treasury/Strat_ReduceReserveLP.sol -> 57-58

```
1         // 2. swap Weth -> YToken
2         swap(WethUtils.weth.balanceOf(address(this)), 0);
```

LOCATION	treasury/Strat_ReduceReserveLP.sol -> 71-72

```
1         WethUtils.weth.safeIncreaseAllowance(address(swapRouter),
  _wethToSwap);
2
   swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(_weth
  ToSwap, _minYTokenOut, swapPaths, address(this), block.timestamp +
  SWAP_TIMEOUT);
```

DESCRIPTION	The minimum amount supplied to the *swap* function is *0* which means that this swap will succeed no matter how little is received. This creates the potential for someone to front-run the swap by manipulating the price and ensuring very little is received during the swap. Note that while *SwapStrategyPOL* does add a minimum token amount out, the calculation of slippage should be done off-chain to save gas.
RECOMMENDATION	Pass in the expected minimum amount from an off-chain caller or calculate the appropriate minimum using a time-weighted price oracle.
RESOLUTION	The *Strat_ReduceReserveLP* contract now takes a minimum token amount as a parameter when reducing the reserve.

No emergencyWithdraw Function

FINDING ID	#0009
SEVERITY	Medium Risk
STATUS	Open
LOCATION	farms/FantasticStaking.sol

DESCRIPTION	There is no *emergencyWithdraw* function. Rev-3 The function *exit* was renamed to *emergencyWithdraw* There is no function that will allow the user to withdraw funds in an emergency. The current function requires that *getReward* completes successfully, which in theory could fail.
RECOMMENDATION	Add an additional function that allows the user to claim their deposited balance without *getReward* succeeding.
RESOLUTION	N/A

Minters Addresses Cannot Easily Be Identified

FINDING ID	#0010
SEVERITY	Low Risk
STATUS	Partially Mitigated
LOCATION	• FantasticStaking.sol -> 55: *mapping(address => bool) public minters;*

DESCRIPTION	The minters' addresses noted are not easily identified by users. It is important to ensure that all addresses are accounted for.
RECOMMENDATION	Add an enumerated list of all minters addresses. Add events that emit when an address is added or removed.
RESOLUTION	Fixed in revision - 5 but the deployed contract is using an old version. The minters could be checked by examining the constructor arguments on the block explorer. New minters can not be added after deployment.

Setters Could Cause Deployment Issues

FINDING ID	#0011
SEVERITY	Low Risk
STATUS	Partially Closed
LOCATION	• FantasticChef.sol -> 292-295: *function setRewardMinter(IFantasticStaking _rewardMinter) external* • FsmReserve.sol -> 38-43: *function setRewarder(address _rewarder) external returns (bool)* • FsmReserve.sol -> 45-50: *function setDaoFund(address _daoFund) external returns (bool)* • FsmReserve.sol -> 52-56: *function setTreasuryFund(address _treasuryFund) external* • FsmReserve.sol -> 58-63: *function setPool(address _pool) external returns (bool)* • XToken.sol -> 26-29: *function setMinter(address _minter) external*

DESCRIPTION	These functions can be called by anyone. Once it is set it can not further be changed. A malicious actor could wrongfully set it during deployment before the actual owner has set it. causing contracts to have to be redeployed.
RECOMMENDATION	Add *onlyOwner* modifier to the functions.
RESOLUTION	Some of the functions have been removed from FsmReserve.sol. Project team comment: "We accept the risk to reduce the level of Centralized control"

Use Safe Transfer For Native Token

FINDING ID	#0012
SEVERITY	Low Risk
STATUS	Partially Closed
LOCATION	• FantasticStaking.pol -> 349: *payable(msg.sender).transfer(reward);* • Pool.sol -> 326: *payable(_sender).transfer(_ethAmount);*

DESCRIPTION	Direct transfer functions are called.
RECOMMENDATION	For the transfer of native token payable modifier should use *call* with a reentrancy guard instead of *address.transfer*. https://docs.soliditylang.org/en/v0.8.4/security-considerations.html#sending-and-receiving-ether
RESOLUTION	Deployed contract: built-in transfer is still used in FantasticStaking. Fixed in revision - 5 but not deployed. The project team now uses `sendValue` instead.

Spot Price Normalized To 18 Decimals

FINDING ID	#0013
SEVERITY	Low Risk
STATUS	Partially Closed
LOCATION	libs/SpotPriceGetter.sol -> 22-29

```
        uint8 _token0MissingDecimals = 18 -
(ERC20(_token0).decimals());
        uint8 _token1MissingDecimals = 18 -
(ERC20(_token1).decimals());
        uint256 _price = 0;
        if (_token == _token0) {
            _price =
_reserve1.mul(10**_token1MissingDecimals).mul(PRICE_PRECISION).div(_res
erve0);
        } else {
            _price =
_reserve0.mul(10**_token0MissingDecimals).mul(PRICE_PRECISION).div(_res
erve1);
        }
```

LOCATION	oracles/UniswapPairOracle.sol -> 77-84

```
        uint8 _token0MissingDecimals = 18 -
(ERC20(_token0).decimals());
        uint8 _token1MissingDecimals = 18 -
(ERC20(_token1).decimals());
        uint256 _price = 0;
        if (token == _token0) {
            _price =
_reserve1.mul(10**_token1MissingDecimals).mul(pricePrecision).div(_rese
rve0);
        } else {
            _price =
_reserve0.mul(10**_token0MissingDecimals).mul(pricePrecision).div(_rese
rve1);
        }
```

DESCRIPTION	Token prices are normalized to use 18 decimal places. This can lead to unexpected calculation results in contracts that use the spot price. Note that the oracle TWAP price does not normalize the price at all.

RECOMMENDATION	Remove the price normalization.
RESOLUTION	*libs/SpotPriceGetter.sol* contract was removed. *oracles/UniswapPairOracle.sol* is still normalized to 18 decimals.

Unconventional Burn Method

FINDING ID	#0014
SEVERITY	Low Risk
STATUS	Closed
LOCATION	XToken.sol -> 85-90

```
1    function burn(address _from, uint256 _amount) external onlyMinter {
2        _burn(_from, _amount);
3    }
```

DESCRIPTION	The XToken minter is allowed to burn from any address without approval from the address. Note: burn typically refers to burning from msg.sender (own balance) while burnFrom refers to burning from another address.
RECOMMENDATION	Add an approval check in the burn function.
RESOLUTION	The burn function was removed.

Unbounded Loop

FINDING ID	#0015
SEVERITY	Low Risk
STATUS	Partially Closed
LOCATION	Looping over *rewardTokens.length*: • farms/FantasticStaking.sol -> 149-155: *for (uint256 i = 0; i < _rewards.length; i++) {* • farms/FantasticStaking.sol -> 342-355: *for (uint256 i; i < rewardTokens.length; i++) {* • farms/FantasticStaking.sol -> 447-455: *for (uint256 i = 1; i < rewardTokens.length; i++) {* Looping over *userEarnings[msg.sender]*: • farms/FantasticStaking.sol -> 168-173: *for (uint256 i = 0; i < earnings.length; i++) {* • farms/FantasticStaking.sol -> 229-236: *for (uint256 i = 0; i < length; i++) {* • farms/FantasticStaking.sol -> 307-327: *for (uint256 i = 0; ; i++) {* Looping over *userLocks[user]*: • farms/FantasticStaking.sol -> 208-219: *for (uint256 i = 0; i < locks.length; i++) {* • farms/FantasticStaking.sol -> 385-389: *for (uint256 i = 0; i < length; i++) {* Looping over *poolInfo.length*: • FantasticChef.sol -> 87-89:*for (uint256 i = 0; i < len; ++i)* • FantasticChef.sol -> 226-228: *for (uint256 pid = 0; pid < length; ++pid)* • FantasticChef.sol -> 235-237: *for (uint256 pid = 0; pid < length; ++pid)*

DESCRIPTION	Iterating over an unbounded array can cause transactions to revert due to the gas limit. Note that the loops make frequent use of *delete* when lock and earning entries are no longer required. This does not automatically shift the array to replace the deleted entry.

RECOMMENDATION	Provide a limit to the size of the array. Alternatively, pass a lower and upper index as parameters and iterate over a range.
RESOLUTION	The following unbound loops still remain after fix: Looping over *rewardTokens.length*: ● farms/FantasticStaking.sol -> 149-155: *for (uint256 i = 0; i < _rewards.length; i++) {* ● farms/FantasticStaking.sol -> 342-355: *for (uint256 i; i < rewardTokens.length; i++) {* ● farms/FantasticStaking.sol -> 447-455: *for (uint256 i = 1; i < rewardTokens.length; i++) {* Looping over *userEarnings[msg.sender]*: ● farms/FantasticStaking.sol -> 168-173: *for (uint256 i = 0; i < earnings.length; i++) {* ● farms/FantasticStaking.sol -> 229-236: *for (uint256 i = 0; i < length; i++) {* ● farms/FantasticStaking.sol -> 307-327: *for (uint256 i = 0; ; i++) {* Looping over *userLocks[user]*: ● farms/FantasticStaking.sol -> 208-219: *for (uint256 i = 0; i < locks.length; i++) {* ● farms/FantasticStaking.sol -> 385-389: *for (uint256 i = 0; i < length; i++) {*

Comment Does Not Match Ether Allocation

FINDING ID	#0016
SEVERITY	Informational
STATUS	Mitigated
LOCATION	FsmDaoFund.sol -> 9

```
1    uint256 public constant ALLOCATION = 3_000_000 ether; // 20%
```

LOCATION	FsmTreasuryFund.sol -> 9

```
1    uint256 public constant ALLOCATION = 3_000_000 ether; // 10%
```

LOCATION	FSM.sol -> 8

```
1    uint256 public constant MAX_TOTAL_SUPPLY = 30_000_000 ether;
```

DESCRIPTION	The comment in FsmDaoFund does not match the actual percentage. Rev-5: The *ALLOCATION* was updated to return 6_000_000 to match the comment but only 3_000_000 was minted to it. A proxy contract is deployed that will return the true amount of allocation 3_000_000.
RECOMMENDATION	Update the comment to match the allocation.
RESOLUTION	This is fixed on the deployed contract but GitHub is/was not updated at the time of audit.

rewardDistributors Addresses Cannot Easily Be Identified

FINDING ID	#0017
SEVERITY	Informational
STATUS	Open
LOCATION	• FantasticStaking.sol -> 57: *mapping(address => mapping(address => bool)) public rewardDistributors;*

DESCRIPTION	The rewardDistributors addresses noted are not easily identified by users. It is important to ensure that all addresses are accounted for.
RECOMMENDATION	Add an enumerated list of all rewardDistributors addresses. Add events that emit when an address is added or removed.
RESOLUTION	The function now emits an event when a rewardDistributor is added. The addresses are still not easily identified by users.

Swaps always execute on the same block

FINDING ID	#0018
SEVERITY	Informational
STATUS	Closed
LOCATION	treasury/Strat_ReduceReserveLP.sol#70-73

```
1     function swap(uint256 _wethToSwap, uint256 _minYTokenOut) internal
  {
2         WethUtils.weth.safeIncreaseAllowance(address(swapRouter),
  _wethToSwap);
3
   swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(_weth
  ToSwap, _minYTokenOut, swapPaths, address(this), block.timestamp +
  SWAP_TIMEOUT);
4     }
```

DESCRIPTION	The smart contract calls always execute in the same block in which they are triggered therefore adding additional time to *block.timestamp* is unnecessary. The swap will always be complete by *block.timestamp*.
RECOMMENDATION	Remove + *SWAP_TIMEOUT*.
RESOLUTION	The project team has implemented the recommended fix.

SafeMath Unnecessary In Solidity 0.8

FINDING ID	#0019
SEVERITY	Informational
STATUS	Partially Closed
LOCATION	SpotPriceGetter.sol -> 11 UniswapPairOracle.sol -> 13 FantasticStaking.sol -> 16

```
1    using SafeMath for uint256;
```

DESCRIPTION	SafeMath is no longer necessary in solidity 0.8.
RECOMMENDATION	Remove SafeMath.
RESOLUTION	SafeMath is still used in FantasticStaking.sol.

Initial FSM Distribution

FINDING ID	#0024
SEVERITY	High Risk
STATUS	Mitigated
LOCATION	Rev-3 - tokens/fantom/FSM.sol -> 15

```
1         _mint(msg.sender, maxTotalSupply());
```

DESCRIPTION	The total supply FSM in revisions 1 and 2 was split with a small amount for initial liquidity, 2 larger amounts for the DAO and treasury funds, and the remainder in the reserve. This was changed so that the entire maximum supply of the FSM token is minted to the contract deployer.
RECOMMENDATION	Ensure the initial token distribution is split between the appropriate addresses.
RESOLUTION	Tokens will be minted directly to the appropriate fund and treasury addresses. The amount minted to the FsmDaoFund does not match the expected distribution. This is fixed on the deployed contract but GitHub is/was not updated at the time of audit.

SwapStrategyPOL Slippage Calculation

FINDING ID	#0025
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	Rev-3 - SwapStrategyPOL.sol -> 15

```
1             uint256 _minYTokenOut = (yTokenAmt * (SLIPPAGE_PRECISION -
  swapSlippage)) / SLIPPAGE_PRECISION;
2             uint256 _minWethOut = (wethAmt * (SLIPPAGE_PRECISION -
  swapSlippage)) / SLIPPAGE_PRECISION;
3             yToken.safeIncreaseAllowance(address(swapRouter),
  yTokenAmt);
4             WethUtils.weth.safeIncreaseAllowance(address(swapRouter),
  wethAmt);
5             (uint256 _amountA, uint256 _amountB, uint256 _liquidity) =
  swapRouter.addLiquidity(
6                 address(yToken),
7                 address(WethUtils.weth),
8                 yTokenAmt,
9                 wethAmt,
10                _minYTokenOut,
11                _minWethOut,
12                treasury,
13                block.timestamp
14            );
```

DESCRIPTION	The calculation of minimum tokens when adding liquidity uses the balance of the swap strategy contract. This can result in adding liquidity reverting on the following conditions: • Y tokens or weth are sent to the contract, causing the balances to never match that of the liquidity pool • The price impact of the initial swap is too high This will in turn prevent users from minting new x tokens.
RECOMMENDATION	Pass in the expected minimum amount from an off-chain caller or calculate the appropriate minimum using a time-weighted price oracle. Ensure that any discrepancy between the y token and wETH balances does not prevent adding liquidity. This may be partially mitigated by replacing the swap

	strategy if it malfunctions.
RESOLUTION	The project team has implemented the recommended fix. The amounts of *wETH* and *yToken* are passed in as arguments rather than using the current contract balances.

Use safeTransfer for ERC20 transfer

FINDING ID	#0026
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	• FsmDaoFund.sol -> 42-49: *function transfer(address _receiver, uint256 _amount) external onlyOwner* • FsmTreasuryFund.sol -> : 42-49: *function transfer(address _receiver, uint256 _amount) external onlyOwner*

DESCRIPTION	ERC20 transfers do not revert by default on failure. This can lead to loss of funds or other unexpected behaviors.
RECOMMENDATION	Use the OpenZeppelin *SafeERC20* methods *safeTransfer* and *safeTransferFrom* for ERC20 transfers.
RESOLUTION	The project team has implemented the recommended fix.

Whitelisted Addresses Can Bypass Flashloan Protection

FINDING ID	#0027
SEVERITY	High Risk
STATUS	Closed
LOCATION	Rev-4 - Pool.sol -> 306-309:

```
function zapCollect() external {
        require(whitelist[msg.sender], "Pool::zapCollect: Only allow senders from whitelist");
        doCollect(msg.sender);
    }
```

DESCRIPTION	Whitelisted addresses are able to bypass the flashloan protection which prevents claiming rewards within the same block as a mint or redeem action. The project team has indicated that this is to allow for flexibility when using Zap contracts. However, this re-introduces the ability to manipulate prices and drain the zapper's rewards.
RECOMMENDATION	Do not bypass the flashloan protection mechanism. Allow users to claim zap swap rewards as though they minted or redeemed themselves.
RESOLUTION	Whitelist and zapCollect have been removed.

Static Analysis

Function Should Be Marked As View

FINDING ID	#0020
SEVERITY	Informational
STATUS	Closed
LOCATION	• YToken.sol -> 10: *function maxTotalSupply() internal virtual returns (uint256);*

DESCRIPTION	Functions that do not modify the state should be marked as a view.
RECOMMENDATION	Change the function to a view function.
RESOLUTION	The project team has implemented the recommended fix.

Unused Library

FINDING ID	#0021
SEVERITY	Informational
STATUS	Closed
LOCATION	libs/Math.sol

DESCRIPTION	Local copy of OpenZepplines Math is never used.
RECOMMENDATION	Remove the unused library.
RESOLUTION	Library was removed.

No Events Emitted For Changes To Protocol Values

FINDING ID	#0022
SEVERITY	Informational
STATUS	Closed
LOCATION	• oracles/UniswapPairOracle.sol -> 40-42: *function setPeriod(uint256 _period) external onlyOwner {* • farms/FantasticStaking.sol -> 94-100: *function addReward(address _rewardsToken, address _distributor) public onlyOwner {* • farms/FantasticStaking.sol -> 103-107: *function approveRewardDistributor(address _rewardsToken, address _distributor, bool _approved) external onlyOwner {* • Pool.sol -> 423-426: *function setTreasury(address _treasury) 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.

Unused Event

FINDING ID	#0023
SEVERITY	Informational
STATUS	Closed
LOCATION	Strat_ReduceReserveLP.sol -> 77 SwapStrategyPOL.sol -> 103

```
1     event SwapConfigUpdated(address indexed _router, uint256 _slippage,
  address[] _paths);
```

DESCRIPTION	The noted events are never used.
RECOMMENDATION	Remove the events.
RESOLUTION	The project team implemented the recommended fix.

On-Chain Analysis

Timelock Delay Is Short

FINDING ID	#0028
SEVERITY	High Risk
STATUS	Open
LOCATION	FantasticChef 0x9c09eA872582bA02E0008C4853eAA5199bF8D0a7 Pool 0xa3B99CdFdDe2216AfB1D58D6108cC93fea413A76 FantasticTreasury 0xcF53c4d2d4A5452a60240cfd87543d6D6b7Faf53 MasterOracle 0x1c430Bc0A4AD0c360B4eA5A3147dc079035b6E68 UniswapPairOracle - YToken 0xf324c3A156C653fC7a1810f10404A48F88FEe100 UniswapPairOracle - XToken 0x024B7c6f6f80bf893236ac96543d15FD2545cD50

DESCRIPTION	The timelock contract has a 12-hour delay. Obelisk recommends a delay of at least 72 hours.
RECOMMENDATION	Increase the timelock delay or switch to the 72-hour timelock.
RESOLUTION	N/A

Timelock Minimum Delay Is Short

FINDING ID	#0029
SEVERITY	Medium Risk
STATUS	Open
LOCATION	Timelock - 12 hours 0xA505FB8292C103a77756e02f4Be334e863969dDe Timelock - 72 hours 0xcA27Df2B8A6e9A69A092867c06E4B8583975e483

DESCRIPTION	The timelock minimum delay is set to 1 hour. The current delay on timelocks is 12 hours and 72 hours.
RECOMMENDATION	Ensure that the timelock delay is not reduced to the new minimum.
RESOLUTION	N/A

FXM Treasury Fund Owner is an Unverified Contract

FINDING ID	#0030
SEVERITY	Medium Risk
STATUS	Open
LOCATION	FXM Treasury Fund 0x30eC57fb7DdaC4EF08aDD2DE675504F63d13C167

DESCRIPTION	Functions that do not modify the state should be marked as a view. The *owner* of the FXM Treasury Fund is an unverified contract. The functionality of this contract has not been examined.
RECOMMENDATION	Make the *owner* of the Treasury Fund a verified Timelock or Multisig contract.
RESOLUTION	N/A

Initial Slippage is Out Of Bounds

FINDING ID	#0031
SEVERITY	Medium Risk
STATUS	Mitigated
LOCATION	Pool.sol -> 387-389:

```
    if (yTokenSlippage == PRECISION) {
        // ignore slipapge between Twap and Spot
        return;
    }
```

LOCATION	Pool.sol -> 493-500:

```
    function setYTokenSlippage(uint256 _slippage) external onlyOwner {
        require(
            _slippage <= 300000,
            "Pool::setYTokenSlippage: yTokenSlippage cannot be more than 30%"
        );
        yTokenSlippage = _slippage;
        emit YTokenSlippageSet(_slippage);
    }
```

LOCATION	Pool 0xa3B99CdFdDe2216AfB1D58D6108cC93fea413A76

DESCRIPTION	When *yTokenSlippage* is equal to *PRECISION* there is no slippage protection applied. This means that swaps may be front-run and funds lost. Note: that once *yTokenSlippage* is set to a value below *30%*, it cannot be set to *PRECISION* again. The value of *yTokenSlippage* has been set to 20%.
RECOMMENDATION	No change is required.
RESOLUTION	The value of *yTokenSlippage* has already been set to below the 30% limit.

TWAP Price Normalized To 18 Decimals

FINDING ID	#0032
SEVERITY	Low Risk
STATUS	Mitigated
LOCATION	oracles/UniswapPairOracle.sol -> 79-90:

```
    uint8 _token0MissingDecimals = 18 - (ERC20(token0).decimals());
    uint8 _token1MissingDecimals = 18 - (ERC20(token1).decimals());
    if (token == token0) {
        amountOut =
            price0Average.mul(pricePrecision).decode144() *
            (10**_token1MissingDecimals);
    } else {
        require(token == token1, "PairOracle: INVALID_TOKEN");
        amountOut =
            price1Average.mul(pricePrecision).decode144() *
            (10**_token0MissingDecimals);
    }
```

LOCATION	UniswapPairOracle - YToken 0xf324c3A156C653fC7a1810f10404A48F88FEe100 UniswapPairOracle - XToken 0x024B7c6f6f80bf893236ac96543d15FD2545cD50

DESCRIPTION	The oracles were modified to return TWAP normalized to 18 decimal places. This can lead to unexpected behavior if the oracle is used in other systems where tokens do not have 18 decimals. The noted oracles should not be redeployed using tokens with other than 18 decimals.
RECOMMENDATION	No change is required.
RESOLUTION	XToken and YToken already have 18 decimals.

Minor Changes To Contracts

FINDING ID	#0033
SEVERITY	Informational
STATUS	Closed
LOCATION	FTMX 0x75Ab56F4eFE81598a78EF3079a331F1D0336765D FXM 0x132b56763C0e73F95BeCA9C452BadF89802ba05e FantasticChef 0x9c09eA872582bA02E0008C4853eAA5199bF8D0a7 FxmReserve 0x9CB3a75FaA95d91021F1B499eD3dC8D366f27eE9 FxmDevFund 0x37676771DDDd1109153Cc07eC50f7E2461823909 FxmDaoFundFixed 0x4EF8E70DeBe3a5ACdC066556996B3CA9d47FFc81

DESCRIPTION	The noted contracts have the following changes: XFTM.sol • XFTM renamed to FTMX. • Minter check moved to a modifier. FSM.sol • FSM renamed to FXM. • maxTotalSupply moved to YToken. FantasticChef.sol • Max pools reduced to 24 FsmReserve.sol • FsmReserve renamed to FxmReserve Fund.sol • Renamed field Fund.fsm to Fund.ytoken

	FsmDevFund.sol ● Renamed FsmDevFund to FxmDevFund. FsmDaoFund.sol ● Renamed FsmDaoFund to FxmDaoFund. FsmTreasuryFund.sol ● Renamed FsmTreasuryFund to FxmTreasuryFund.
RECOMMENDATION	No changes are necessary.
RESOLUTION	N/A

External Addresses

Externally Owned Accounts

Owner

ACCOUNT	0x9362e8cF30635de48Bdf8DA52139EEd8f1e5d400
USAGE	0x37676771DDDd1109153Cc07eC50f7E2461823909 *FxmDevFund.owner* - Variable 0xA505FB8292C103a77756e02f4Be334e863969dDe *Timelock.admin* - Variable 0xcA27Df2B8A6e9A69A092867c06E4B8583975e483 *Timelock.admin* - Variable 0xeb76144D08854eCd7500de9EF968E81D455713Dd *SwapStrategyPOL.owner* - Variable
IMPACT	• receives elevated permissions as owner, operator, or other

External Contracts

These contracts are not part of the audit scope.

SpookySwap Router

ADDRESS	0xF491e7B69E4244ad4002BC14e878a34207E38c29
USAGE	0xeb76144D08854eCd7500de9EF968E81D455713Dd *SwapStrategyPOL.swapRouter* - Immutable
IMPACT	• ERC20 Token

FundWithdrawal

ADDRESS	0xF491e7B69E4244ad4002BC14e878a34207E38c29
USAGE	0x30eC57fb7DdaC4EF08aDD2DE675504F63d13C167 *SwapStrategyPOL.swapRouter* - Immutable
IMPACT	• receives transfer of tokens deposited or minted by project

External Tokens

These contracts are not part of the audit scope.

Wrapped FTM

ADDRESS	0x21be370D5312f44cB42ce377BC9b8a0cEF1A4C83
USAGE	0xC4510604504Fd50f64499fF6186AEf1F740dE38B *FantasticStaking.rewardTokens[1]* - Variable, no setter UniswapPairOracle - XToken 0x024B7c6f6f80bf893236ac96543d15FD2545cD50 *UniswapPairOracle.token0* - Immutable UniswapPairOracle - YToken 0xf324c3A156C653fC7a1810f10404A48F88FEe100 *UniswapPairOracle.token1* - Immutable
IMPACT	• ERC20 Token

WFTM-FTMX SpookySwap LP

ADDRESS	0x215c8E1452681be980Bce575cF719029581Ef263
USAGE	UniswapPairOracle - XToken 0x024B7c6f6f80bf893236ac96543d15FD2545cD50 *UniswapPairOracle.pair* - Immutable 0x9c09eA872582bA02E0008C4853eAA5199bF8D0a7 *FantasticChef.lpToken[1]* - Variable, no setter
IMPACT	• Uniswap LP Token

FXM-WFTM SpookySwap LP

ADDRESS	0x664D417B404404268C4E571975B4eC77157B8aC4
USAGE	0xf324c3A156C653fC7a1810f10404A48F88FEe100 *SwapStrategyPOL.swapRouter* - Immutable 0x9c09eA872582bA02E0008C4853eAA5199bF8D0a7 *FantasticChef.lpToken[0]* - Variable, no setter

IMPACT	• Uniswap LP Token

Appendix A - Reviewed Documents

Deployed Contracts

Document	Address
farms/FantasticChef.sol	0x9c09eA872582bA02E0008C4853eAA5199bF8D0a7
farms/FantasticStaking.sol	0xC4510604504Fd50f64499fF6186AEf1F740dE38B
funds/FsmDaoFund.sol	Initial deployment: 0x0A7cABf40e293dd795cD052798369451aDee910e Fixed deployment 0x4EF8E70DeBe3a5ACdC066556996B3CA9d47FFc81 also deployed as FxmDevFund 0x37676771DDDd1109153Cc07eC50f7E2461823909
funds/FsmReserve.sol	renamed to FxmReserve 0x9CB3a75FaA95d91021F1B499eD3dC8D366f27eE9
funds/FsmTreasuryFund.sol	renamed to FxmTreasuryFund 0x30eC57fb7DdaC4EF08aDD2DE675504F63d13C167
oracles/MasterOracle.sol	0x1c430Bc0A4AD0c360B4eA5A3147dc079035b6E68
oracles/UniswapPairOracle.sol	UniswapPairOracle - XToken 0x024B7c6f6f80bf893236ac96543d15FD2545cD50 UniswapPairOracle - YToken 0xf324c3A156C653fC7a1810f10404A48F88FEe100
tokens/fantom/FSM.sol	renamed to FXM 0x132b56763C0e73F95BeCA9C452BadF89802ba05e
tokens/fantom/XFTM.sol	renamed to FTMX 0x75Ab56F4eFE81598a78EF3079a331F1D0336765D
treasury/FantasticTreasury.sol	0xcF53c4d2d4A5452a60240cfd87543d6D6b7Faf53
treasury/Strat_Recollateralize.sol	N/A
treasury/Strat_ReduceReserveLP.sol	N/A
Pool.sol	0xa3B99CdFdDe2216AfB1D58D6108cC93fea413A76

SwapStrategyPOL.sol	0xeb76144D08854eCd7500de9EF968E81D455713Dd
Timelock.sol	Timelock - 12 hours 0xA505FB8292C103a77756e02f4Be334e863969dDe Timelock - 72 hours 0xcA27Df2B8A6e9A69A092867c06E4B8583975e483

Libraries and Interfaces

interfaces/IFantasticStaking.sol interfaces/IFantasticTreasury.sol interfaces/IMasterOracle.sol interfaces/IPairOracle.sol interfaces/IPool.sol interfaces/IRewarder.sol interfaces/ISwapStrategy.sol interfaces/IWETH.sol interfaces/IXToken.sol interfaces/IYToken.sol interfaces/IYTokenReserve.sol libs/Babylonian.sol libs/FixedPoint.sol libs/Math.sol libs/SpotPriceGetter.sol libs/UQ112x112.sol libs/WethUtils.sol libs/MasterOracle.sol libs/UniswapPairOracle.sol tokens/XToken.sol tokens/YToken.sol

Revisions

Revision 1	5fb8fb2946b4daea0e49d5a8ffe5313aa88f5755
Revision 2	<skipped>
Revision 3	59ff53f033f69fe57b089dc392db00dcc27f872b
Revision 4	3a32796e57f7520bc8dc6e9e5153ed6e7372df56
Revision 5	ce9d4b1187103191e77586f5aafb8e0c1f610c14

Imported Contracts

OpenZeppelin	4.5.0

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