Sushi Miso contest
Findings & Analysis Report

2021-11-05

Table of contents

Overview

About C4

Code 432n4 (C4) is an open organization consisting of security researchers, auditors, developers, and individuals with domain expertise in smart contracts.

A C4 code contest is an event in which community participants, referred to as Wardens, review, audit, or analyze smart contract logic in exchange for a bounty provided by sponsoring projects.

During the code contest outlined in this document, C4 conducted an analysis of the Sushi Miso contest smart contract system written in Solidity. The code contest took place between September 9—September 15 2021.

Wardens

11 Wardens contributed reports to the Sushi Miso contest code contest:

- WatchPug
- 0xRajeev
- cmichel
- gpersoon
- JMukesh
- leastwood
- hrkrshnn
- pauliax
- itsmeSTYJ
- loop

This contest was judged by ghoul.sol.

Final report assembled by itsmetechjay and CloudEllie.

Summary

The C4 analysis yielded an aggregated total of 25 unique vulnerabilities and 101 total findings. All of the issues presented here are linked back to their original finding.

Of these vulnerabilities, 3 received a risk rating in the category of HIGH severity, 1 received a risk rating in the category of MEDIUM severity, and 21 received a risk rating in the category of LOW severity.

C4 analysis also identified 47 non-critical recommendations and 29 gas optimizations.

Scope

The code under review can be found within the C4 Sushi Miso contest repository and is composed of 106 smart contracts written in the Solidity programming language, and includes 4,040 lines of Solidity code.

Severity Criteria

C4 assesses the severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into three primary risk categories: high, medium, and low.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Further information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on the C4 website.

High Risk Findings (3)

[H-01] `PostAuctionLauncher.sol#finalize()` Adding liquidity to an existing pool may allows the attacker to steal most of the tokens

Submitted by WatchPug, also found by 0xRajeev and cmichel.

`PostAuctionLauncher.finalize()` can be called by anyone, and it sends tokens directly to the pair pool to mint liquidity, even when the pair pool exists.

An attacker may control the LP price by creating the pool and then call `finalize()` to mint LP token with unfair price (pay huge amounts of tokens and get few amounts of LP token), and then remove the initial liquidity they acquired when creating the pool and take out huge amounts of tokens.

https://github.com/sushiswap/miso/blob/2cdb1486a55ded55c81898b7be8811cb68cfda9e/contracts/Liquidity/PostAuctionLauncher.sol#L257

```
    /**
     * @notice Finalizes Token sale and launches LP.
     * @return liquidity Number of LPs.
     */
    function finalize() external nonReentrant returns (uint256 liqui
        // GP: Can we remove admin, let anyone can finalise and laur
        // require(hasAdminRole(msg.sender) || hasOperatorRole(msg.s
        require(marketConnected(), "PostAuction: Auction must have t
        require(!launcherInfo.launched);

        if (!market.finalized()) {
            market.finalize();
        }
        require(market.finalized());

        launcherInfo.launched = true;
        if (!market.auctionSuccessful() ) {
            return 0;
        }

        /// @dev if the auction is settled in weth, wrap any contrac
```

```
        uint256 launcherBalance = address(this).balance;
        if (launcherBalance > 0 ) {
            IWETH(weth).deposit{value : launcherBalance}();
        }

        (uint256 token1Amount, uint256 token2Amount) =  getTokenAmou

        /// @dev cannot start a liquidity pool with no tokens on eit
        if (token1Amount == 0 || token2Amount == 0 ) {
            return 0;
        }

        address pair = factory.getPair(address(token1), address(toke
        if(pair == address(0)) {
            createPool();
        }

        /// @dev add liquidity to pool via the pair directly
        _safeTransfer(address(token1), tokenPair, token1Amount);
        _safeTransfer(address(token2), tokenPair, token2Amount);
        liquidity = IUniswapV2Pair(tokenPair).mint(address(this));
        launcherInfo.liquidityAdded = BoringMath.to128(uint256(launc

        /// @dev if unlock time not yet set, add it.
        if (launcherInfo.unlock == 0 ) {
            launcherInfo.unlock = BoringMath.to64(block.timestamp +
        }
        emit LiquidityAdded(liquidity);
    }
```

In line 257, `PostAuctionLauncher` will mint LP with `token1Amount` and `token2Amount`. The amounts (`token1Amount` and `token2Amount`) are computed according to the auction result, without considering the current price (reserves) of the existing `tokenPair`.

See PostAuctionLauncher.getTokenAmounts()

`PostAuctionLauncher` will receive an unfairly low amount of lp token because the amounts sent to `tokenPair` didn't match the current price of the pair.

See UniswapV2Pair.mint(...)

```
liquidity = MathUniswap.min(amount0.mul(_totalSupply) / _reserve
```

Impact

Lose a majority share of the tokens.

Proof of Concept

1. The attacker creates LP with 0.0000001 token1 and 1000 token2, receives 0.01 LP token;
2. Call `PostAuctionLauncher.finalize()`. PostAuctionLauncher will mint liquidity with 2000 token1 and 1000 token2 for example, receives only 0.01 LP token;
3. The attacker removes all his LP, receives 1000 token1 (most of which come from `PostAuctionLauncher`).

Recommended Mitigation Steps

To only support tokenPair created by `PostAuctionLauncher` or check for the token price before mint liquidity.

Clearwood (Sushi Miso) confirmed and patched:

> https://github.com/sushiswap/miso/pull/21

[H-02] SushiToken transfers are broken due to wrong delegates accounting on transfers

Submitted by cmichel.

When minting / transferring / burning tokens, the `SushiToken._beforeTokenTransfer` function is called and supposed to correctly shift the voting power due to the increase/decrease in tokens for the `from` and `to` accounts. However, it does not correctly do that, it tries to shift the votes from the `from` account, instead of the `_delegates[from]` account. This can lead to transfers reverting.

Proof Of Concept

Imagine the following transactions on the `SushiToken` contract. We'll illustrate the corresponding `_moveDelegates` calls and written checkpoints for each.

- `mint(A, 1000) = transfer(0, A, 1000)` => `_moveDelegates(0, delegates[A]=0)` => no checkpoints are written to anyone because delegatees are still zero
- A delegates to A' => `_moveDelegates(0, A')` => `writeCheckpoint(A', 1000)`
- B delegates to B' => no checkpoints are written as B has a zero balance
- `transfer(A, B, 1000)` => `_moveDelegates(A, delegates[B] = B')` => underflows when subtracting `amount=1000` from A's non-existent checkpoint (defaults to 0 votes)

It should subtract from A's delegatee `A'` 's checkpoint instead.

Impact

Users that delegated votes will be unable to transfer any of their tokens.

Recommended Mitigation Steps

In `SushiToken._beforeTokenTransfer` , change the `_moveDelegates` call to be from `_delegates[from]` instead:

```
function _beforeTokenTransfer(address from, address to, uint256
    _moveDelegates(_delegates[from], _delegates[to], amount);
    super._beforeTokenTransfer(from, to, amount);
}
```

This is also how the [original code from Compound] does it.

[maxsam4 (Sushi Miso) acknowledged]:

> This is a known issue in Sushi token but was kept unchanged in MISO for "preservation of history :)". That was not necessarily a wise choice lol. I think 1 severity should be fine for this as this was an intentional thing. The delegate

> feature is not supposed to be used in these tokens. We might create a new token type with this bug fixed.

ghoul-sol (judge) commented:

> We have crazy wallets on the blockchain that will call every possible function available to them and that's why I'm keeping this as is. Even intentional, the issue stands so the warden should get credit for it.

[H-03] Last person to withdraw his tokens might not be able to do this, in Crowdsale (edge case)

Submitted by gpersoon.

Impact

Suppose a Crowdsale is successful and enough commitments are made before the `marketInfo.endTime`. Suppose marketStatus.commitmentsTotal == marketInfo.totalTokens -1 // note this is an edge case, but can be constructed by an attacker Then the function `auctionEnded()` returns true Assume `auctionSuccessful()` is also true (might depend on the config of `marketPrice.goal` and `marketInfo.totalTokens`) Then an admin can call `finalize()` to finalize the Crowdsale. The function finalize distributes the funds and the unsold tokens and sets `status.finalized = true` so that finalized cannot be called again. Now we have "marketInfo.totalTokens -1" tokens left in the contract

However `commitEth()` or `commitTokens()` can still be called (they give no error message that the auction has ended) Then functions call `calculateCommitment`, which luckily prevent from buying too much, however 1 token can still be bought These functions also call `\_addCommitment()`, which only checks for `marketInfo.endTime`, which hasn't passed yet.

Now an extra token is sold and the contract has 1 token short. So the last person to withdraw his tokens cannot withdraw them (because you cannot specify how much you want to withdraw)

Also the revenues for the last token cannot be retrieved as `finalize()` cannot be called again.

Proof of Concept

https://github.com/sushiswap/miso/blob/master/contracts/Auctions/Crowdsale.sol#L374

```
 function finalize() public nonReentrant {
        require(hasAdminRole(msg.sender) || wallet == msg.sender
        MarketStatus storage status = marketStatus;
        require(!status.finalized, "Crowdsale: already finalized
        MarketInfo storage info = marketInfo;
        require(auctionEnded(), "Crowdsale: Has not finished yet

        if (auctionSuccessful()) {
            /// @dev Transfer contributed tokens to wallet.
            /// @dev Transfer unsold tokens to wallet.
        } else {
            /// @dev Return auction tokens back to wallet.
        }
        status.finalized = true;

function auctionEnded() public view returns (bool) {
        return block.timestamp > uint256(marketInfo.endTime) ||
        _getTokenAmount(uint256(marketStatus.commitmentsTotal) +
    }

function auctionSuccessful() public view returns (bool) {
        return uint256(marketStatus.commitmentsTotal) >= uint256
}

function commitEth(address payable _beneficiary, bool readAndAgr
       ...
        uint256 ethToTransfer = calculateCommitment(msg.value);
       ...
       _addCommitment(_beneficiary, ethToTransfer);

 function calculateCommitment(uint256 _commitment) public view r
        uint256 tokens = _getTokenAmount(_commitment);
        uint256 tokensCommited =_getTokenAmount(uint256(marketSt
        if ( tokensCommited.add(tokens) > uint256(marketInfo.tot
            return _getTokenPrice(uint256(marketInfo.totalTokens
        }
        return _commitment;
    }

function _addCommitment(address _addr, uint256 _commitment) inte
```

```
        require(block.timestamp >= uint256(marketInfo.startTime)
        ...
        uint256 newCommitment = commitments[_addr].add(_commitme
        ...
        commitments[_addr] = newCommitment;

function withdrawTokens(address payable beneficiary) public   nc
        if (auctionSuccessful()) {
            ...
            uint256 tokensToClaim = tokensClaimable(beneficiary)
            ...
            claimed[beneficiary] = claimed[beneficiary].add(toke
            _safeTokenPayment(auctionToken, beneficiary, tokensT
        } else {



## Tools Used

## Recommended Mitigation Steps
In the function _addCommitment, add a check on auctionEnded() or
```

Clearwood (Sushi Miso) confirmed and patched:

> https://github.com/sushiswap/miso/pull/20

Medium Risk Findings (1)

[M-01] use of transfer() instead of call() to send eth

Submitted by JMukesh.

Impact

Use of `transfer()` might render ETH impossible to withdraw because after istanbul hardfork, there is an increase in the gas cost of the SLOAD operation and therefore breaks some existing smart contracts.Those contracts will break because their fallback functions used to consume less than 2300 gas, and they'll now consume more, since 2300 the amount of gas a contract's fallback function receives if it's called via Solidity's `transfer()` or `send()` methods. Any smart contract that uses

`transfer()` or `send()` is taking a hard dependency on gas costs by forwarding a fixed amount of gas: 2300.

- https://consensys.net/diligence/blog/2019/09/stop-using-soliditys-transfer-now/
- https://blog.openzeppelin.com/opyn-gamma-protocol-audit/

Proof of Concept

- https://github.com/sushiswap/miso/blob/2cdb1486a55ded55c81898b7be8811cb68cfda9e/contracts/MISOTokenFactory.sol#L242
- https://github.com/sushiswap/miso/blob/2cdb1486a55ded55c81898b7be8811cb68cfda9e/contracts/MISOMarket.sol#L256
- https://github.com/sushiswap/miso/blob/2cdb1486a55ded55c81898b7be8811cb68cfda9e/contracts/MISOLauncher.sol#L251
- https://github.com/sushiswap/miso/blob/2cdb1486a55ded55c81898b7be8811cb68cfda9e/contracts/MISOFarmFactory.sol#L244

Tools Used

manual review

Recommended Mitigation Steps

use `call()` to send eth

maxsam4 (Sushi Miso) disputed and commented:

> This is intentional, not a risk. The contract does not want to give any gas stipend to the destination.

> Even if the user messes up, `misoDev` address can be changed to a proper address later.

ghoul-sol (judge) commented:

> using `.transfer` can make ETH transfer to a smart contract impossible. User can always change the address however I agree with warden that this is an issue.

Low Risk Findings (21)

- [L-01] Outdated and Vulnerable `TimelockController.sol` Contract *Submitted by leastwood, also found by JMukesh.*
- [L-02] Frontrunning Initialization of Contracts *Submitted by leastwood.*
- [L-03] Event parameters interchanged for emit of access control template addition *Submitted by 0xRajeev.*
- [L-04] `TokenVault` incorrectly tracks `userIndex` *Submitted by cmichel.*
- [L-05] funds will get lost in deployAccessControl if devaddr isn't set *Submitted by gpersoon.*
- [L-06] An adversarial attacker can initialize ListFactory *Submitted by hrkrshnn.*
- [L-07] The first escrow index underflows *Submitted by pauliax.*
- [L-08] MISORecipe01 uses outdated interfaces *Submitted by pauliax.*
- [L-09] Certain view functions should be used only by UI and not by the code
- [L-10] Front-running cancelAuction can prevent auction cancellation *Submitted by 0xRajeev.*
- [L-11] Usage of address.transfer _Submitted by cmichel, also found by 0xRajeev.
- [L-12] deployMarket may revert due to integer underflow from missing threshold check *Submitted by 0xRajeev.*
- [L-13] Init functions are susceptible to front-running *Submitted by 0xRajeev.*
- [L-14] Loss of price precision *Submitted by cmichel, also found by itsmeSTYJ and leastwood.*
- [L-15] `MISOMasterChef` may not be used with fee-on-transfer tokens *Submitted by cmichel.*
- [L-16] No ERC20 safe* versions called in MisoRecipe *Submitted by cmichel.*
- [L-17] No ERC20 `safeApprove` versions called *Submitted by cmichel.*
- [L-18] finalize() can be succesfully called before initMarket() *Submitted by gpersoon.*
- [L-19] `currentTemplateId` is Not Actively Removed by `MISOLauncher.removeLiquidityLauncherTemplate()` *Submitted by leastwood.*

- [L-20] lockTokens should validate withdrawer *Submitted by pauliax.*
- [L-21] Payable external init is redundant and may allow unaccounted token claims or DoS *Submitted by 0xRajeev.*

Non-Critical Findings (47)

- [N-01] Missing useful isOpen() function could save gas *Submitted by 0xRajeev.*
- [N-02] Inaccurate Function Name `enableList()` *Submitted by leastwood.*
- [N-03] Unused imports *Submitted by pauliax.*
- [N-04] Missing zero-address check on beneficiary may lead to loss of funds *Submitted by 0xRajeev.*
- [N-05] Single-step wallet address change is risky *Submitted by 0xRajeev.*
- [N-06] Same LP token can be added more than once to affect reward calculations *Submitted by 0xRajeev.*
- [N-07] excessive eth is not transfered back to the deployer if msg.value is greater than minimum fees *Submitted by JMukesh.*
- [N-08] Lack of Factory Contract for `TokenList.sol` *Submitted by leastwood.*
- [N-09] Tokens without 18 decimals are unhandled *Submitted by 0xRajeev.*
- [N-10] Critical withdrawTokens function is missing an event *Submitted by 0xRajeev.*
- [N-11] Missing zero-address checks *Submitted by 0xRajeev.*
- [N-12] Missing Events on State Changing Functions *Submitted by leastwood, also found by pauliax and 0xRajeev.*
- [N-13] Missing contract existence check may cause silent failures of token transfers *Submitted by 0xRajeev.*
- [N-14] Relying on setters for initialisation of critical parameters is risky *Submitted by 0xRajeev.*
- [N-15] Lack of indexed event parameters will affect offchain monitoring *Submitted by 0xRajeev.*
- [N-16] Unused event may be unused code or indicative of missed emit/logic *Submitted by 0xRajeev.*
- [N-17] Lack of Input Validation *Submitted by leastwood, also found by 0xRajeev, cmichel, and JMukesh.*

- [N-18] TokenInitialized token parameter is always empty *Submitted by pauliax, also found by 0xRajeev.*
- [N-19] Unconventional use of basis points for integratorFeePct could cause undefined behavior *Submitted by 0xRajeev.*
- [N-20] Old Solidity compiler version *Submitted by 0xRajeev.*
- [N-21] `AccessControlTemplateRemoved` event not used *Submitted by cmichel.*
- [N-22] Should `TokenList` implement `IPointList`? *Submitted by cmichel.*
- [N-23] Use constant named variable for auction decimals *Submitted by cmichel.*
- [N-24] `HyperbolicAuction.initAuction`'s `_factor` argument is never used *Submitted by cmichel.*
- [N-25] `MISOMasterChef.setDevPercentage` should be capped *Submitted by cmichel.*
- [N-26] Commitments can happen after already finalized *Submitted by cmichel.*
- [N-27] Unused event `StrategyCvxHelper.HarvestState` *Submitted by cmichel.*
- [N-28] Requiring a decimals method for ERC-20 tokens is non-standard *Submitted by hrkrshnn.*
- [N-29] Teams should be warned not to accept rebasing tokens as payment currencies *Submitted by itsmeSTYJ.*
- [N-30] Divide Before Multiply *Submitted by leastwood.*
- [N-31] `_safeApprove()` is Not Used Instead of `approve()` *Submitted by leastwood.*
- [N-32] Unchecked `fundsCommitted` in Token Withdrawal *Submitted by leastwood.*
- [N-33] PostAuctionLauncher _deposit require condition contradicts error message *Submitted by loop.*
- [N-34] _addCommitment should check that address is not empty *Submitted by pauliax.*
- [N-35] Consider using a solidity version >= 0.8.0
- [N-36] Add input validation on some methods

- [N-37] Use a struct for raw data.
- [N-38] use of floating pragma *Submitted by JMukesh.*
- [N-39] comment copy paste error *Submitted by gpersoon, also found by itsmeSTYJ, leastwood, and loop.*
- [N-40] Typo in comment in PointList.sol *Submitted by itsmeSTYJ.*
- [N-41] Improper Boolean Comparison *Submitted by leastwood.*
- [N-42] Missing `uint256` Cast *Submitted by leastwood.*
- [N-43] Inconsistent Template Deletion *Submitted by leastwood.*
- [N-44] Missing SPDX Identifier *Submitted by leastwood.*
- [N-45] Inclusive checks *Submitted by pauliax.*
- [N-46] Style issues *Submitted by pauliax.*
- [N-47] getTokenTemplate should check boundaries *Submitted by pauliax.*

Gas Optimizations (29)

- [G-01] Slot packing saves slots but increases runtime gas consumption due to masking *Submitted by 0xRajeev.*
- [G-02] Caching state variables in local/memory variables avoids SLOADs to save gas *Submitted by 0xRajeev.*
- [G-03] Avoiding initialization of loop index can save a little gas *Submitted by 0xRajeev.*
- [G-04] Check for zero msg.value can save gas *Submitted by 0xRajeev.*
- [G-05] Using function parameters in emits saves gas *Submitted by 0xRajeev.*
- [G-06] Avoiding unnecessary external call will save > 2600 gas *Submitted by 0xRajeev.*
- [G-07] Unnecessary zero check on variable which is never initialized earlier *Submitted by 0xRajeev.*
- [G-08] unused local variable *Submitted by JMukesh.*
- [G-09] Gas: Cache auction prices *Submitted by cmichel, also found by leastwood.*
- [G-10] Gas: Remove nonce from parameter list *Submitted by cmichel.*
- [G-11] gas improvement in isInList *Submitted by gpersoon.*

- [G-12] Upgrade to at least 0.8.4 *Submitted by hrkrshnn.*
- [G-13] ## Caching the length in for loops *Submitted by hrkrshnn.*
- [G-14] Use `calldata` instead of `memory` for function parameters *Submitted by hrkrshnn.*
- [G-15] Consider having short revert strings *Submitted by hrkrshnn.*
- [G-16] Caching `totalPoints` during `setPoints` method *Submitted by hrkrshnn.*
- [G-17] Redundant _newAddress parameter for deprecateFactory *Submitted by itsmeSTYJ.*
- [G-18] Unnecessary addition in finalize() function *Submitted by itsmeSTYJ.*
- [G-19] Redundant liquidityAdded check *Submitted by itsmeSTYJ.*
- [G-20] Lack of `Immutable` Keyword *Submitted by leastwood.*
- [G-21] Consolidation of Storage Slots *Submitted by leastwood.*
- [G-22] cancelAuction function is public, but not called internally *Submitted by loop.*
- [G-23] Require statement in PostAuctionLauncher finalize() function will never be reached. *Submitted by loop.*
- [G-24] Separate minter roles are not really necessary *Submitted by pauliax.*
- [G-25] Useless initialization to default value *Submitted by pauliax.*
- [G-26] Dead code *Submitted by pauliax.*
- [G-27] allDepositIds is pretty much useless *Submitted by pauliax.*
- [G-28] Pack structs tightly *Submitted by pauliax.*
- [G-29] *startTime is always < 10000000000 when _endTime < 10000000000 (*endTime > _startTime) *Submitted by pauliax.*

Disclosures

C4 is an open organization governed by participants in the community.

C4 Contests incentivize the discovery of exploits, vulnerabilities, and bugs in smart contracts. Security researchers are rewarded at an increasing rate for finding higher-risk issues. Contest submissions are judged by a knowledgeable security researcher and solidity developer and disclosed to sponsoring developers. C4 does not conduct

formal verification regarding the provided code but instead provides final verification.

C4 does not provide any guarantee or warranty regarding the security of this project. All smart contract software should be used at the sole risk and responsibility of users.

Top