



Cyberscope

Audit Report

Cultiv8

May 2025

Repository <https://github.com/Cv8Org/Contracts/commits/feat/latest-fixes/>

Commit [2c41f64d652eb4859595c221d1c5b5cf991d0a29](https://github.com/Cv8Org/Contracts/commit/2c41f64d652eb4859595c221d1c5b5cf991d0a29)

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	3
Audit Updates	3
Source Files	3
Overview	5
Findings Breakdown	6
Diagnostics	7
CO - Code Optimization	9
Description	9
Recommendation	11
CR - Code Repetition	12
Description	12
Recommendation	13
CCR - Contract Centralization Risk	14
Description	14
Recommendation	15
DPI - Decimals Precision Inconsistency	16
Description	16
Recommendation	16
Team Update	17
MEE - Missing Events Emission	18
Description	18
Recommendation	20
MSC - Missing Sanity Check	21
Description	21
Recommendation	22
Team Update	22
PIL - Potential Incorrect Logic	23
Description	23
Recommendation	24
PIMTC - Potential Incorrect Max Transaction Calculation	25
Description	25
Recommendation	26
PVC - Price Volatility Concern	27
Description	27
Recommendation	28
RCS - Redundant Code Segments	29
Description	29
Recommendation	29

RSW - Redundant Storage Writes	30
Description	30
Recommendation	30
RSD - Redundant Swap Duplication	31
Description	31
Recommendation	31
UPV - Unbound Protocol Values	32
Description	32
Recommendation	33
Team Update	33
L04 - Conformance to Solidity Naming Conventions	34
Description	34
Recommendation	35
L13 - Divide before Multiply Operation	36
Description	36
Recommendation	36
L17 - Usage of Solidity Assembly	37
Description	37
Recommendation	37
Functions Analysis	38
Inheritance Graph	49
Flow Graph	50
Summary	51
Disclaimer	52
About Cyberscope	53

Review

Repository	https://github.com/Cv8Org/Contracts/commits/feat/latest-fixes/
Commit	2c41f64d652eb4859595c221d1c5b5cf991d0a29

Audit Updates

Initial Audit	08 May 2024
Corrected Phase 2	02 Aug 2024
Corrected Phase 3	16 Feb 2025
Corrected Phase 4	12 May 2025
Corrected Phase 5	22 May 2025
Corrected Phase 6	26 May 2025

Source Files

Filename	SHA256
LotteryToken.sol	7b837fa5a4a4e64e7cbe15ccf4b29ffa6537 6d571037b9b763e2c3f1ad22d012
lib/PancakeAdapter.sol	be3f7c5d0e5da813dfb1d78498873e217a 2901691c17cf383f17edef121f9566
lib/ConstantsAndTypes.sol	55e7218eed94fced7507e45f3c7102b94df d9a82a618e8d8f46ec228505e5cb0
lib/configs/VRFConsumerConfig.sol	1943326b87e19a2776da08fe25d840aa1f7 861ef7b0cfabd07c6a34d4eb782a3

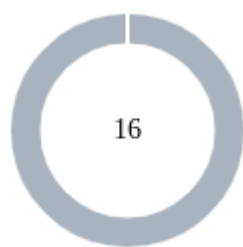
lib/configs/ProtocolConfig.sol	ea3a3f2624d8fe8191dfe339cbfda5276b0 58e0ec91a42bf0d57bf42c432e5f3
lib/configs/LotteryEngineConfig.sol	4af3576a410b76a27f777221d518249bee4 00b82e36d6e76171623e25e976fbe
lib/configs/Configuration.sol	5695ef2e5c4d1f111768d51c476a1903162 2f41e1e28930af5649a1cd100dc89

Overview

Cultiv8 introduces a reflection token incorporating a lottery mechanism powered by Chainlink VRF. The lottery system comprises three distinct types:

1. **Smash Time:** Triggered when a buy or sell transaction meets or exceeds the `significantAmount` threshold (currently set to 0.1 USD in settings). For every multiple of this amount, users receive one ticket, up to a maximum of 10 tickets per transaction. The winning ticket is drawn from all participants after the event concludes, with the prize distributed based on Chainlink randomness.
2. **Donation:** Participation in the donation lottery requires sending a minimum amount to the designated donation address. The quantity of donation tickets correlates with the total donation amount. Additionally, owners can generate donation tickets. This lottery type is initiated upon any transfer if sufficient eligible addresses are present.
3. **Holder:** Triggered by any transfer, the holder lottery awards tickets based on the number of transactions occurring since the last lottery event. Participants receive up to 3 tickets depending on their holding amount.

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	16

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	0	16	0	0

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	RSW	Redundant Storage Writes	Acknowledged
●	RCS	Redundant Code Segments	Acknowledged
●	MEE	Missing Events Emission	Acknowledged
●	CCR	Contract Centralization Risk	Acknowledged
●	MSC	Missing Sanity Check	Acknowledged
●	CO	Code Optimization	Acknowledged
●	CR	Code Repetition	Acknowledged
●	UPV	Unbound Protocol Values	Acknowledged
●	PIMTC	Potential Incorrect Max Transaction Calculation	Acknowledged
●	RSD	Redundant Swap Duplication	Acknowledged
●	DPI	Decimals Precision Inconsistency	Acknowledged
●	PIL	Potential Incorrect Logic	Acknowledged
●	PVC	Price Volatility Concern	Acknowledged
●	L04	Conformance to Solidity Naming Conventions	Acknowledged

●	L13	Divide before Multiply Operation	Acknowledged
●	L17	Usage of Solidity Assembly	Acknowledged

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L1310,1378
Status	Acknowledged

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The function `_addDonationsLotteryTickets` contains a redundant conditional check towards the end of the function to verify if donations lottery is enabled and if the number of unique donators exceeds the minimum required for participation. However, this check is unnecessary since the function already begins by verifying if the donations lottery is enabled.

Furthermore, the `uniqueDonatorsCounter` could be checked only after the variable has been incremented.

Finally, the `uniqueDonatorsCounter` is updated within a for loop, that could be optimized by using a local variable and assigning it to the `uniqueDonatorsCounter` variable after the end of the for loop.

```
function _addDonationsLotteryTickets(address _transferrer, address
_recipient, uint256 _amount) private {
    if (!_lotteryConfig.donationsLotteryEnabled) {
        return;
    }
    // if this transfer is a donation, add a ticket for transferrer.
    if (_recipient == _lotteryConfig.donationAddress && _amount >=
_lotteryConfig.minimalDonation) {
        if (_donatorTicketIdxs[_donationRound][_transferrer].length == 0)
        {
            uniqueDonatorsCounter++;
        }
        uint256 length = _donators.length;
        _donators.push(_transferrer);
        _donatorTicketIdxs[_donationRound][_transferrer].push(length);
    }
    if (_lotteryConfig.donationsLotteryEnabled && uniqueDonatorsCounter
>= _lotteryConfig.minimumDonationEntries) {
        _donationsLottery();
    }
}
...
function mintDonationTickets(address[] calldata _recipients, uint256[]
calldata _amounts) external onlyOwner {
    uint256 recipientsLength = _recipients.length;
    if (recipientsLength != _amounts.length) {
        revert RecipientsLengthNotEqualToAmounts();
    }

    uint256 round = _donationRound;
    for (uint256 i = 0; i < recipientsLength;) {
        address recipient = _recipients[i];
        uint256 amount = _amounts[i];
        uint256 idx = _donatorTicketIdxs[round][recipient].length;
        uint256 newIdx = idx + amount;

        if (_donatorTicketIdxs[round][recipient].length == 0) {
            _uniqueDonatorsCounter++;
        }

        for (; idx < newIdx;) {
            _donators.push(recipient);
            _donatorTicketIdxs[round][recipient].push(idx);

            unchecked {
                ++idx;
            }
        }

        unchecked {
```

```
        ++i;  
    }  
}  
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

CR - Code Repetition

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L827,856
Status	Acknowledged

Description

During the assessment, we identified redundant logic in the contract related to prize conversion for two different lotteries, namely the SmashTime Lottery and the Donation Lottery. Specifically, the contract contains two private methods named `_convertSmashTimeLotteryPrize` and `_convertDonationLotteryPrize`, both of which essentially perform the same operations with minor differences in variable names.

In both methods, the following steps are executed:

1. Determine the amount of tokens held in the respective lottery prize pool.
2. Transfer these tokens to the contract.
3. Convert the transferred tokens into BNB (Binance Coin) using the `_swapTokensForBNB` function.
4. Update the BNB prize balance for the corresponding lottery.

While the functionality is identical, the only discrepancies lie in the variables used to identify the prize pool address and the BNB prize balance storage variable.

```
function _convertSmashTimeLotteryPrize() private {
    uint256 conversionAmount =
    balanceOf(smashTimeLotteryPrizePoolAddress);

    _tokenTransfer(smashTimeLotteryPrizePoolAddress, address(this),
    conversionAmount);

    uint256 convertedBNB = _swapTokensForBNB(conversionAmount);
    unchecked {
        smashtimeLotteryBNBPrize += convertedBNB;
    }
}

/**
 * Convert prize for Donation Lottery.
 */
function _convertDonationLotteryPrize() private {
    uint256 conversionAmount =
    balanceOf(donationLotteryPrizePoolAddress);

    _tokenTransfer(donationLotteryPrizePoolAddress, address(this),
    conversionAmount);

    uint256 convertedBNB = _swapTokensForBNB(conversionAmount);
    unchecked {
        donationLotteryBNBPrize += convertedBNB;
    }
}
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L1365,1470,1487,1507,1522,1526,1530,1534,1538,1542,1547,1551 contracts/lib/Configuration.sol#L53,59,63,67,73,77,83,89,95,99,103,107,111,115,119,123,127,131,135,139,143,147,151,155,159,168,172,176,185
Status	Acknowledged

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function mintDonationTickets(address[] calldata _recipients, uint256[]  
calldata _amounts) external onlyOwner  
...  
function updateHolderList(address[] calldata _holdersToCheck) external  
onlyOwner  
...  
function updateHolderList(address[] calldata _holdersToCheck) external  
onlyOwner  
...  
function includeInReward(address _account) external onlyOwner  
...  
function setWhitelist(address _account, bool _status) external onlyOwner  
...  
function setMaxBuyPercent(uint256 _maxBuyPercent) external onlyOwner  
...  
function setSwapAndLiquifyEnabled(bool _enabled) external onlyOwner  
...  
function setLiquiditySupplyThreshold(uint256 _amount) external onlyOwner  
...  
function setFeeSupplyThreshold(uint256 _amount) external onlyOwner  
...  
function setThreeDaysProtection(bool _enabled) external onlyOwner  
...  
function withdraw(uint256 _amount) external onlyOwner  
...  
function withdrawBNB(uint256 _amount) external onlyOwner
```

...

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

DPI - Decimals Precision Inconsistency

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L1353
Status	Acknowledged

Description

The decimals field of a contract's ERC20 token can be used to specify the number of decimal places that the token uses. For example, if decimals are set to `8`, it means that the smallest unit of the token is `0.00000001`, and if decimals are set to `18`, it means that the smallest unit of the token is `0.00000000000000000001`.

However, there is an inconsistency in the way that the decimals field is handled in some ERC20 contracts. The ERC20 specification does not specify how the decimals field should be implemented, and as a result, some contracts use different precision numbers.

This inconsistency can cause problems when interacting with these contracts, as it is not always clear how the decimals field should be interpreted. For example, if a contract expects the decimals field to be 18 digits, but the contract being interacted with uses 8 digits, the result of the interaction may not be what was expected.

```
uint256 usdAmount = _TokenPriceInUSD(_amount) / 1e18;  
uint256 hundreds = usdAmount / 100;
```

Recommendation

To avoid these issues, it is important to carefully review the implementation of the decimals field of the underlying tokens. The team is advised to normalize each decimal to one single source of truth. A recommended way is to scale all the decimals to the greatest token's decimal. Hence, the contract will not lose precision in the calculations.

The following example depicts 3 tokens with different decimals precision.

ERC20	Decimals
Token 1	6
Token 2	9
Token 3	18

All the decimals could be normalized to 18 since it represents the ERC20 token with the greatest digits.

Team Update

The team states that the address under review (0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56) corresponds to BUSD, which uses 18 decimals, as confirmed on [BscScan](#).

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L555,1499,1516,1522,1526,1530,1534,1538,1542 contracts/lib/configs/Configuration.sol
Status	Acknowledged

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function _takeLiquidity(uint256 _rLiquidity, uint256 _tLiquidity) private
...
function excludeFromReward(address _account) public onlyOwner
...
function includeInReward(address _account) external onlyOwner
...
function setWhitelist(address _account, bool _status) external onlyOwner
...
function setMaxBuyPercent(uint256 _maxBuyPercent) external onlyOwner
...
function setSwapAndLiquifyEnabled(bool _enabled) external onlyOwner
...
function setLiquiditySupplyThreshold(uint256 _amount) external onlyOwner
...
function setFeeSupplyThreshold(uint256 _amount) external onlyOwner
...
function setThreeDaysProtection(bool _enabled) external onlyOwner
...
function setHolderLotteryPrizePoolAddress(
    address _newAddress
) external onlyOwner
...
function setSmashTimeLotteryPrizePoolAddress(
    address _newAddress
) external onlyOwner
...
function setDonationLotteryPrizePoolAddress(
    address _newAddress
) external onlyOwner
...
function setTeamAddress(address _newAddress) external onlyOwner
...
function setTeamAccumulationAddress(address _newAddress) external
onlyOwner
...
function setTreasuryAddress(address _newAddress) external onlyOwner
...
function setTreasuryAccumulationAddress(address _newAddress) external
onlyOwner
...
function setFeeConfig(uint256 _feeConfigRaw) external onlyOwner
...
function switchSmashTimeLotteryFlag(bool flag) external onlyOwner
...
function switchHoldersLotteryFlag(bool flag) external onlyOwner
...
function switchDonationsLotteryFlag(bool flag) external onlyOwner
...
function excludeFromFee(address account) external onlyOwner
...
```

```
function includeInFee(address account) external onlyOwner
...
function setHoldersLotteryTxTrigger(uint64 _txAmount) external onlyOwner
...
function setHoldersLotteryMinPercent(uint256 _minPercent) external
onlyOwner
...
function setDonationAddress(address _donationAddress) external onlyOwner
...
function setMinimalDonation(uint256 _minimalDonation) external onlyOwner
...
function setFee(uint256 _fee) external onlyOwner
...
function setMinimumDonationEntries(uint64 _minimumEntries) external
onlyOwner
...
function setSmashTimePrizePercent(uint256 _value) external onlyOwner
...
function setHoldersLotteryPrizePercent(uint256 _value) external
onlyOwner
...
function setDonationLotteryPrizePercent(uint256 _value) external
onlyOwner
...
function setTreasuryPlainTokenPercent(uint256 _value) external onlyOwner
...
function setSignificantAmountForTransfer(uint256 _value) external
onlyOwner
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L187,1526,1534,1538 contracts/lib/Configuration.sol#L53
Status	Acknowledged

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The function arguments are not properly sanitized.

```
constructor (
    address _mintSupplyTo,
    address _coordinatorAddress,
    address _routerAddress,
    address _wnnbAddress,
    address _tUSDAddress,
    uint256 _fee,
    ConsumerConfig memory _consumerConfig,
    DistributionConfig memory _distributionConfig,
    LotteryConfig memory _lotteryConfig
)
    VRFConsumerBaseV2(_coordinatorAddress)
    PancakeAdapter(_routerAddress, _wnnbAddress, _tUSDAddress)
    Configuration(_fee, _consumerConfig, _distributionConfig,
_lotteryConfig)
{
    ...
    function setMaxBuyPercent(uint256 _maxBuyPercent) external onlyOwner
    ...
    function setLiquiditySupplyThreshold(uint256 _amount) external
onlyOwner
    ...
    function setFeeSupplyThreshold(uint256 _amount) external onlyOwner
    ...
    function setFeeConfig(uint256 _feeConfigRaw) external onlyOwner
    ...
    function setFee(uint256 _fee) external onlyOwner
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

Team Update

The team added a check to `setMaxBuyPercent` function.

PIL - Potential Incorrect Logic

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L960
Status	Acknowledged

Description

The contract implements a mechanism to differentiate between regular transfers and other types of transactions, such as those involving liquidity pools. The relevant code segment for this check is as follows:

```
// if pair is not involved => its a regular transfer
bool regularTransfer = (_sender != PANCAKE_PAIR && _recipient !=
PANCAKE_PAIR) ||
    (_isExcludeFromRegularTransfer[_sender] ||
_isExcludeFromRegularTransfer[_recipient]);
```

The logic intends to mark a transaction as a regular transfer if both the sender and recipient are not the liquidity pair address (`PANCAKE_PAIR`). However, it incorrectly includes a condition that checks if either the sender or recipient is marked as excluded from regular transfers (`_isExcludeFromRegularTransfer`).

The boolean logic within this condition is flawed due to the use of the `or` (`||`) operator, which incorrectly results in the `regularTransfer` flag being set to `true` even when either the sender or recipient is excluded from regular transfers. This contradicts the intended behavior implied by the naming of the `_isExcludeFromRegularTransfer` array.

Specifically, the statement: `solidity`

```
(_isExcludeFromRegularTransfer[_sender] ||
_isExcludeFromRegularTransfer[_recipient])
```

 causes `regularTransfer` to be `true` if either `_sender` or `_recipient` is excluded from regular transfers, regardless of their involvement with the liquidity pair. This can lead to misclassifying transactions involving excluded addresses as regular transfers.

Recommendation

To align the logic with the intended functionality, the condition should be revised to correctly reflect the exclusion logic. The corrected logic should ensure that a transaction is marked as a regular transfer only if neither the sender nor the recipient is the pair address and neither is excluded from regular transfers.

PIMTC - Potential Incorrect Max Transaction Calculation

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L584,906
Status	Acknowledged

Description

The contract includes a mechanism to enforce a maximum transaction limit, ensuring that no single transfer exceeds a predefined percentage of the total supply. The contract also calculates and imposes fees on token transfers, which are intended to be factored into the max transaction limit check. However, the implementation incorrectly assumes the fee will be applied to the current transaction, potentially leading to inaccurate validation.

In the `_antiAbuse` function, the contract attempts to prevent large transactions by comparing the transaction amount against a percentage of the total supply. The relevant code snippet is:

```
uint256 lastUserBalance = balanceOf(_to) + ((_amount * (PRECISION -  
_calcFeePercent())) / PRECISION);  
...  
uint256 allowedAmount = (tSupply * dayLimit) / PRECISION;  
if (lastUserBalance >= allowedAmount) {  
    revert TransferAmountExceededForToday(); // @audit lastUserBalance =  
    allowed amount should not revert  
}
```

Here, the `_amount` is adjusted by subtracting the calculated fee percentage (`_calcFeePercent()`). This adjusted amount is then compared against the allowable max transaction limit. However, the fee might not be applicable to the current transaction due to various conditions (e.g., addresses being excluded from fees), leading to a discrepancy between the actual transaction amount and the assumed amount used in the validation.

As a result, transactions that should be permitted might be incorrectly reverted, and transactions that should be restricted might bypass the limitation.

Recommendation

The team is advised to refactor the `_antiAbuse` function to correctly account for the actual transaction amount after fees are applied, ensuring accurate enforcement of the max transaction limit. This can be achieved by determining the fee applicability before performing the max transaction limit check.

This adjustment ensures that the actual amount being transferred (after fees) is used to check against the max transaction limit, thereby aligning the logic with the intended behavior.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	LotteryToken.sol#L658,1643
Status	Acknowledged

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `liquiditySupplyThreshold` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
uint256 contractTokenBalance = balanceOf(address(this));
    if (
        contractTokenBalance >= liquiditySupplyThreshold &&
        _lock == SwapStatus.Open &&
        _from != PANCAKE_PAIR &&
        _to != PANCAKE_PAIR &&
        swapAndLiquifyEnabled
    ) {
        _swapAndLiquify(contractTokenBalance);
    }

    // distribute fees to team and treasury
    if (_lock == SwapStatus.Open) {
        _distributeFees();
    }

    ...
    function setLiquiditySupplyThreshold(uint256 _amount) external
    onlyOwner {
        liquiditySupplyThreshold = _amount;
    }
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RCS - Redundant Code Segments

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L138,139,145,246,253
Status	Acknowledged

Description

The contract is currently containing code segments, that do not provide any actual functionality. Such redundant code segments can lead to confusion and misinterpretation of the contract's purpose and functionality. Moreover, they contribute to unnecessary bloat in the contract, potentially impacting its efficiency and clarity.

```
// TODO: use real value
```

Recommendation

It is recommended to remove these redundant code segments from the contract. Eliminating these non-functional parts will streamline the contract, making it more efficient and easier to comprehend. This action will also reduce the potential for confusion among users and developers who interact with or audit the contract.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L1522,1530,1542
Status	Acknowledged

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
function setWhitelist(address _account, bool _status) external
onlyOwner {
    whitelist[_account] = _status;
}
...
function setSwapAndLiquifyEnabled(bool _enabled) external onlyOwner
{
    swapAndLiquifyEnabled = _enabled;
}
...
function setThreeDaysProtection(bool _enabled) external onlyOwner {
    threeDaysProtectionEnabled = _enabled;
}
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

RSD - Redundant Swap Duplication

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L664,697,698
Status	Acknowledged

Description

The contract contains multiple swap methods that individually perform token swaps and transfer promotional amounts to specific addresses and features. This redundant duplication of code introduces unnecessary complexity and increases dramatically the gas consumption. By consolidating these operations into a single swap method, the contract can achieve better code readability, reduce gas costs, and improve overall efficiency.

```
function _distributeFees() private lockTheSwap {
    _distributeFeeToAddress(teamFeesAccumulationAddress, teamAddress);
    _distributeFeeToAddress(treasuryFeesAccumulationAddress,
treasuryAddress);
    ...
}
...
function _distributeFeeToAddress(address _feeAccumulationAddress, address
_destinationAddress) private {
    ...
    _swapTokensForTUSD(half, _destinationAddress);
    _swapTokensForBNB(accumulatedBalance - half, _destinationAddress);
    ...
}
```

Recommendation

A more optimized approach could be adopted to perform the token swap operation once for the total amount of tokens and distribute the proportional amounts to the corresponding addresses, eliminating the need for separate swaps.

UPV - Unbound Protocol Values

Criticality	Minor / Informative
Location	contracts/LotteryToken.sol#L1526,1534,1538 contracts/lib/configs/ProtocolConfig.sol#L65,69,73
Status	Acknowledged

Description

The contract contains multiple setter functions that allow the owner to update important parameters. Specifically, the functions `setMaxBuyPercent`, `setLiquiditySupplyThreshold`, and `setFeeSupplyThreshold` do not enforce any upper or lower bounds on the values that can be set.

```
function setMaxBuyPercent(uint256 _maxBuyPercent) external onlyOwner
{
    maxBuyPercent = _maxBuyPercent;
}

function setLiquiditySupplyThreshold(uint256 _amount) external
onlyOwner {
    liquiditySupplyThreshold = _amount;
}

function setFeeSupplyThreshold(uint256 _amount) external onlyOwner {
    feeSupplyThreshold = _amount;
}
...
function _setFeeConfig(uint256 _feeConfigRaw) internal {
    _fees = Fee.wrap(_feeConfigRaw);
}

function _setTreasuryPlainTokenPercent(uint256 _value) internal {
    plainTokenPercent = _value;
}

function _setSignificantAmountForTransfer(uint256 _value) internal {
    significantAmount = _value;
}
```

Recommendation

To mitigate these risks, the team is advised to introduce reasonable upper and lower bounds for each of these parameters. This can be done by adding validation logic within each setter function. By implementing these bounds, the contract can ensure that these critical parameters remain within a safe and reasonable range, thereby reducing the risk of misconfiguration and enhancing the overall security and reliability of the contract.

Team Update

The team added a check to `setMaxBuyPercent` function, that it cannot be zero.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	PancakeAdapter.sol#L8,9,11,13,131 LotteryToken.sol#L129,153,257,270,281,291,308,309,310,334,344,357,533,933,934,1205,1303,1427,1454,1455,1508,1518,1527,1564,1582,1606,1615,1624,1639,1643,1650,1654,1658,1662,1666,1672,1676 lib/PancakeAdapter.sol#L8,9,11,13,131 lib/ConstantsAndTypes.sol#L114,130,150,220,243,250,271,272,277,284,305,306,312,313,322,323,329,330 ConstantsAndTypes.sol#L114,130,150,220,243,250,271,272,277,284,305,306,312,313,322,323,329,330
Status	Acknowledged

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address internal immutable _USDT_ADDRESS
address internal immutable _WBNB_ADDRESS
IPancakeRouter02 public immutable PANCAKE_ROUTER
address public immutable PANCAKE_PAIR

function _TokenPriceInUSD(uint256 _amount) internal view returns (uint256
usdAmount) {
    // generate the uniswap pair path of BNB -> USDT
    address[] memory path = new address[] (3);
    path[0] = address(this);
    path[1] = _WBNB_ADDRESS;
    path[2] = _USDT_ADDRESS;

    usdAmount = PANCAKE_ROUTER.getAmountsOut(_amount, path)[2];
}
uint256 _amount
...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions>.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	lib/ConstantsAndTypes.sol#L123,124,125,126,140,142,143,144,145,146 ConstantsAndTypes.sol#L123,124,125,126,140,142,143,144,145,146
Status	Acknowledged

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of precision.

```
prize = (accumulated * uint32(val)) / PRECISION
first = (prize * uint32(val >> 32)) / PRECISION

uint256 prize = (accumulated * uint32(val)) / PRECISION
first = (prize * uint32(val >> 32)) / PRECISION
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	LotteryToken.sol#L921,1191,1205,1302,1303
Status	Acknowledged

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {  
    _words := add(_array, ONE_WORD)  
}  
  
assembly {  
    winnerIdx := mod(mload(_random), 100)  
}  
  
function transfer_unsafe(recipient, amount) {  
    pop(call(gas(), recipient, amount, 0, 0, 0,  
0))  
}  
  
...
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
TestZ	Implementation	VRFConsumerBaseV2, PancakeAdapter, Configuration		
		Public	✓	VRFConsumerBaseV2 PancakeAdapter Configuration
	name	External		-
	symbol	External		-
	totalSupply	External		-
	balanceOf	Public		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-
	tokenFromReflection	Public		-
		External	Payable	-
	_reflectFee	Private	✓	
	_getValues	Private		

	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	_approve	Private	✓	
	_antiAbuse	Private		
	_transfer	Private	✓	swapLockOnPairCall
	_distributeFees	Private	✓	lockTheSwap
	_distributeFeeToAddress	Private	✓	
	_checkForHoldersLotteryEligibility	Private	✓	
	_holdersEligibilityThreshold	Private		
	_checkForHoldersLotteryEligibilities	Private	✓	
	_convertSmashTimeLotteryPrize	Private	✓	
	_convertDonationLotteryPrize	Private	✓	
	_lotteryOnTransfer	Private	✓	
	_requestRandomWords	Private	✓	
	_toRandomWords	Private		
	fulfillRandomWords	Internal	✓	
	_swapAndLiquify	Private	✓	lockTheSwap
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	

	_transferFromExcluded	Private	✓	
	_transferBothExcluded	Private	✓	
	totalFeePercent	External		-
	_finishRound	Private	✓	
	_calculateSmashTimeLotteryPrize	Private		
	_seedTicketsArray	Private		
	_finishSmashTimeLottery	Private	✓	
	_finishHoldersLottery	Private	✓	
	_finishDonationLottery	Private	✓	
	_smashTimeLottery	Private	✓	
	_triggerHoldersLottery	Private	✓	
	_addDonationsLotteryTickets	Private	✓	
	_donationsLottery	Private	✓	
	transferDonationTicket	External	✓	-
	mintDonationTickets	External	✓	onlyOwner
	holdersLotteryTickets	External		-
	donationLotteryTickets	External		-
	donationLotteryTicketsAmountPerDonat or	External		-
	donate	External	✓	-
	availableHoldersLotteryTickets	External		-
	claimHoldersLotteryTickets	External	✓	-
	updateHolderList	External	✓	onlyOwner
	excludeFromReward	Public	✓	onlyOwner

	includeInReward	External	✓	onlyOwner
	setWhitelist	External	✓	onlyOwner
	setMaxBuyPercent	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	External	✓	onlyOwner
	setLiquiditySupplyThreshold	External	✓	onlyOwner
	setFeeSupplyThreshold	External	✓	onlyOwner
	setThreeDaysProtection	External	✓	onlyOwner
	withdraw	External	✓	onlyOwner
	withdrawBNB	External	✓	onlyOwner
TypesHelpers	Library			
	getPrizes	Internal		
	getPrizes	Internal		
	compact	Internal		
	burnFeePercent	Internal		
	liquidityFeePercent	Internal		
	distributionFeePercent	Internal		
	treasuryFeePercent	Internal		
	devFeePercent	Internal		
	smashTimeLotteryPrizeFeePercent	Internal		
	holdersLotteryPrizeFeePercent	Internal		
	donationLotteryPrizeFeePercent	Internal		
	allTickets	Internal		

	addFirst	Internal	✓	
	removeFirst	Internal	✓	
	existsFirst	Internal		
	addSecond	Internal	✓	
	removeSecond	Internal	✓	
	existsSecond	Internal		
	addThird	Internal	✓	
	existsThird	Internal		
	removeThird	Internal	✓	
VRFConsumerConfig	Implementation			
		Public	✓	-
	_setConfig	Internal	✓	
	_setSubscriptionId	Internal	✓	
	_setCallbackGasLimit	Internal	✓	
	_setRequestConfirmations	Internal	✓	
	_setGasPriceKey	Internal	✓	
ProtocolConfig	Implementation			
		Public	✓	-
	_setHolderLotteryPrizePoolAddress	Internal	✓	
	_setSmashTimeLotteryPrizePoolAddress	Internal	✓	
	_setDonationLotteryPrizePoolAddress	Internal	✓	

	_setTeamAddress	Internal	✓	
	_setTeamAccumulationAddress	Internal	✓	
	_setTreasuryAccumulationAddress	Internal	✓	
	_setTreasuryAddress	Internal	✓	
	_setFeeConfig	Internal	✓	
	_setTreasuryPlainTokenPercent	Internal	✓	
	_setSignificantAmountForTransfer	Internal	✓	
LotteryEngineConfig	Implementation			
		Public	✓	-
	_setSmashTimePrizePercent	Internal	✓	
	_setHoldersLotteryPrizePercent	Internal	✓	
	_setDonationLotteryPrizePercent	Internal	✓	
	_switchSmashTimeLotteryFlag	Internal	✓	
	_switchHoldersLotteryFlag	Internal	✓	
	_setHoldersLotteryTxTrigger	Internal	✓	
	_setHoldersLotteryMinPercent	Internal	✓	
	_setDonationAddress	Internal	✓	
	_switchDonationsLotteryFlag	Internal	✓	
	_setMinimanDonation	Internal	✓	
	_setMinimumDonationEntries	Internal	✓	

Configuration	Implementation	IConfiguration, VRFConsumerConfig, ProtocolConfig, LotteryEngineConfig, Ownable		
		Public	✓	VRFConsumerConfig ProtocolConfig LotteryEngineConfig
	_calcFeePercent	Internal		
	setConsumerConfig	External	✓	onlyOwner
	setSubscriptionId	External	✓	onlyOwner
	setCallbackGasLimit	External	✓	onlyOwner
	setRequestConfirmations	External	✓	onlyOwner
	setGasPriceKey	External	✓	onlyOwner
	setHolderLotteryPrizePoolAddress	External	✓	onlyOwner
	setSmashTimeLotteryPrizePoolAddress	External	✓	onlyOwner
	setDonationLotteryPrizePoolAddress	External	✓	onlyOwner
	setTeamAddress	External	✓	onlyOwner
	setTeamAccumulationAddress	External	✓	onlyOwner
	setTreasuryAddress	External	✓	onlyOwner
	setTreasuryAccumulationAddress	External	✓	onlyOwner
	setFeeConfig	External	✓	onlyOwner
	switchSmashTimeLotteryFlag	External	✓	onlyOwner
	switchHoldersLotteryFlag	External	✓	onlyOwner
	switchDonationsLotteryFlag	External	✓	onlyOwner

	excludeFromFee	External	✓	onlyOwner
	includeInFee	External	✓	onlyOwner
	setHoldersLotteryTxTrigger	External	✓	onlyOwner
	setHoldersLotteryMinPercent	External	✓	onlyOwner
	setDonationAddress	External	✓	onlyOwner
	setMinimalDonation	External	✓	onlyOwner
	setFee	External	✓	onlyOwner
	setMinimumDonationEntries	External	✓	onlyOwner
	setSmashTimePrizePercent	External	✓	onlyOwner
	setHoldersLotteryPrizePercent	External	✓	onlyOwner
	setDonationLotteryPrizePercent	External	✓	onlyOwner
	setTreasuryPlainTokenPercent	External	✓	onlyOwner
	setSignificantAmountForTransfer	External	✓	onlyOwner
	burnFeePercent	External		-
	liquidityFeePercent	External		-
	distributionFeePercent	External		-
	treasuryFeePercent	External		-
	devFeePercent	External		-
	smashTimeLotteryPrizeFeePercent	Public		-
	holdersLotteryPrizeFeePercent	Public		-
	donationLotteryPrizeFeePercent	Public		-
	isExcludedFromFee	External		-
	isExcludedFromReward	External		-


	smashTimeLotteryEnabled	External		-
	holdersLotteryEnabled	External		-
	holdersLotteryTxTrigger	External		-
	holdersLotteryMinPercent	External		-
	donationAddress	External		-
	donationsLotteryEnabled	External		-
	minimumDonationEntries	External		-
	minimalDonation	External		-
	subscriptionId	External		-
	callbackGasLimit	External		-
	requestConfirmations	External		-
	gasPriceKey	External		-
IConfiguration	Interface			
	setConsumerConfig	External	✓	-
	setSubscriptionId	External	✓	-
	setCallbackGasLimit	External	✓	-
	setRequestConfirmations	External	✓	-
	setGasPriceKey	External	✓	-
	setTeamAddress	External	✓	-
	setTeamAccumulationAddress	External	✓	-
	setTreasuryAddress	External	✓	-
	setTreasuryAccumulationAddress	External	✓	-

	setFeeConfig	External	✓	-
	switchSmashTimeLotteryFlag	External	✓	-
	switchHoldersLotteryFlag	External	✓	-
	switchDonationsLotteryFlag	External	✓	-
	excludeFromFee	External	✓	-
	includeInFee	External	✓	-
	setHoldersLotteryTxTrigger	External	✓	-
	setHoldersLotteryMinPercent	External	✓	-
	setDonationAddress	External	✓	-
	setMinimalDonation	External	✓	-
	setFee	External	✓	-
	setMinimumDonationEntries	External	✓	-
	burnFeePercent	External		-
	liquidityFeePercent	External		-
	distributionFeePercent	External		-
	treasuryFeePercent	External		-
	devFeePercent	External		-
	smashTimeLotteryPrizeFeePercent	External		-
	holdersLotteryPrizeFeePercent	External		-
	donationLotteryPrizeFeePercent	External		-
	isExcludedFromFee	External		-
	isExcludedFromReward	External		-
	smashTimeLotteryEnabled	External		-

	holdersLotteryEnabled	External	-
	holdersLotteryTxTrigger	External	-
	holdersLotteryMinPercent	External	-
	donationAddress	External	-
	donationsLotteryEnabled	External	-
	minimumDonationEntries	External	-
	minimalDonation	External	-
	subscriptionId	External	-
	callbackGasLimit	External	-
	requestConfirmations	External	-
	gasPriceKey	External	-

Inheritance Graph

Refer to the detailed images available in the GitHub repository linked below:

 InheritanceGraph.png

Flow Graph

Refer to the detailed images available in the GitHub repository linked below:

 FlowGraph.png

Summary

Cultiv8 is an interesting project that has a friendly and growing community. This audit investigates security issues, business logic concerns and potential improvements. The team has acknowledged the findings.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>