AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

TABLE OF CONTENTS

contact@auditmos.com

AUDITMOS

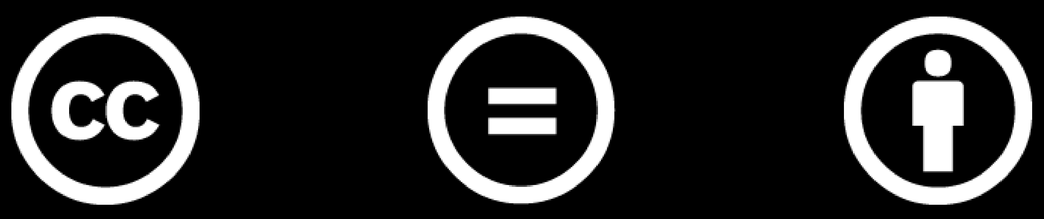
Secure your space

COOKIE3

audit / code review report

June 04, 2024

LICENSE



contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

DISCLAIMER

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

APPROACH AND METHODOLOGY

PURPOSE

1. Determine the correct operation of the protocol, according to the design specification.
2. Identify possible vulnerabilities that could be exploited by an attacker.
3. Detect errors in the smart contract that could lead to unexpected behavior.
4. Analyze whether best practices were followed during development.
5. Make recommendations to improve security and code readability.

CODEBASE

Repository	https://github.com/Cookie3-dev/airdrop-contracts
Branch	master
Commit hash	177cfaddd92bf065456c75953fea7583040b819f

METHODOLOGY

1. Reading the available documentation and understanding the code.
2. Doing automated code analysis and reviewing dependencies.
3. Checking manually source code line by line for security vulnerabilities.
4. Following guildlines and recommendations.
5. Preparing this report.

AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

DESCRIPTION

Issues Categories:

Severity	Description
CRITICAL	vulnerability that can lead to loss of funds, failure to recover blocked funds, or catastrophic denial of service.
HIGH	vulnerability that can lead to incorrect contract state or unpredictable operation of the contract.
MEDIUM	failure to adhere to best practices, incorrect usage of primitives, without major impact on security.
LOW	recommendations or potential optimizations which can lead to better user experience or readability.

Each issue can be in the following state:

State	Description
PENDING	still waiting for resolving
ACKNOWLEDGED	know but not planned to resolve for some reasons
RESOLVED	fixed and deployed

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

AUDIT SCOPE

1. getting to know the project	✓
2. research into architecture	✓
3. manual code read	✓
4. permissions of state changing functions	✓
5. identify common Solidity vulnerabilities	✓
6. test coverage	✓
7. static analysis	✓
8. storage key overlaps	✓
9. DOS possibilities by malicious attacker	✓
10. steal funds possibilities	✓

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

FINDINGS

Finding	Severity	Status
#1 - Possible second preimage attack	MEDIUM	RESOLVED
#2 - The slippage amount is calculated on-chain	LOW	RESOLVED
#3 - The cardinality should be increased after initializing the WETH ↔ Cookie pool	LOW	RESOLVED
#4 - Redundant error	LOW	RESOLVED
#5 - Use msg.sender instead of owner()	LOW	RESOLVED
#6 - The return param from transfer/transferFrom is not handled	LOW	RESOLVED
#7 - Unnecessary checks in the Quoter constructor	LOW	RESOLVED
#8 - Use directly encoded leave instead of hashed proofs	LOW	RESOLVED
#9 - The Swap contract unnecessarily inherits the Ownable contract	LOW	RESOLVED

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

June 04, 2024

#1 - POSSIBLE SECOND PREIMAGE ATTACK

Severity	Status
MEDIUM	RESOLVED

The msg.sender and amount take part in the leaf encoding. These two variables are concatenated and used as leaves in this Merkle tree. msg.sender is of type address, which is equal to 20 bytes and amount is of type uint256, which is 32 bytes. Therefore, their total combined size is 52 bytes. The problem arises when abi.encode is used. abi.encode adds padding when necessary; in our case, it will add 12 bytes before the msg.sender. So, the total number of bytes will be 64, not 52.

Merkle trees that use 64-byte leaves are vulnerable to the second preimage attack. Since internal nodes are 32 byte long (as these are the output of a keccak hash), then these can be combined to prove the presence of certain values that aren't actually leaves in the tree.

RECOMMENDATION

Make the following changes:

Instead of:

```
return keccak256(abi.encode(_account, _amount));
```

Use:

```
return keccak256(abi.encodePacked(_account, _amount));
```

contact@auditmos.com

COOKIE3

audit / code review report

June 04, 2024

#2 - THE SLIPPAGE AMOUNT IS CALCULATED ON-CHAIN

Severity	Status
LOW	RESOLVED

Currently, minAmountOut is calculated on-chain using the TWAP and after that, a 2% fee is deducted. That's the slippage amount which the user will use. If somehow the current price from the TWAP is manipulated by a malicious user, the minAmountOut will be affected.

The most popular solution to calculate the slippage amount is to do it off-chain and pass it as an input parameter directly into the swap function. Of course, considering that we are using the TWAP and that the contracts are deployed on the Base, it is very unlikely for the price to be manipulated. However, for added safety, I would recommend calculating the slippage off-chain.

RECOMMENDATION

The slippage amount should be calculated off-chain. When the user selects the amount in your front-end, deduct a 2% fee. Then, call the quote function from the Quoter contract and use the result as an input parameter to the swap function of the Swapper contract.

AUDITMOS
Secure your space

COOKIE3

audit / code review report

June 04, 2024

#3 - THE CARDINALITY SHOULD BE INCREASED AFTER INITIALIZING THE WETH ↔ COOKIE POOL

Severity	Status
LOW	RESOLVED

Currently, a cardinality of 256 means that there will be 256 slots for observation. So, if there is a price movement in the pool every block, this equals 256 * 13 = 3328 observations, approximately every 55 minutes. Therefore, our TWAP interval of 30 minutes will be satisfied.

Unfortunately, the WETH ↔ Cookie pool will be created on Base and 256 can be problematic. On Base, there is a block every 2 seconds, so 256 * 2 = 512 observations, approximately every 8 minutes. This will affect the precision of the timing and obtaining the correct price

RECOMMENDATION

Increase the cardinality to [1000-1800].

contact@auditmos.com

COOKIE3

audit / code review report

June 04, 2024

#4 - REDUNDANT ERROR

In the AirdropClaim contract, the error TransferFailed is never used.

Severity	Status
LOW	RESOLVED

RECOMMENDATION

Remove the redundant error.

COOKIE3

audit / code review report

June 04, 2024

#5 - USE MSG.SENDER INSTEAD OF OWNER()

Severity	Status
LOW	RESOLVED

When a function is only accessible by the owner due to the presence of the onlyOwner modifier, there is no need to call an internal function to obtain the owner's address. In this case, msg.sender is equal to owner().

RECOMMENDATION

Make the following changes:

Instead of:

```
return i_token.transfer(owner(), i_token.balanceOf(address(this)));
```

Use:

```
return i_token.transfer(msg.sender, i_token.balanceOf(address(this)));
```

COOKIE3

audit / code review report

June 04, 2024

Severity	Status
LOW	RESOLVED

#6 - THE RETURN PARAM FROM TRANSFER/TRANSFERFROM IS NOT HANDLED

In several places in the codebase, the transfer and transferFrom functions are used. Both functions return a boolean to indicate if the transfer is successful. Currently, the return result is not handled.

RECOMMENDATION

Check the return result; if it is false, revert the whole transaction.

contact@auditmos.com

COOKIE3

audit / code review report

June 04, 2024

#7 - UNNECESSARY CHECKS IN THE QUOTER CONSTRUCTOR

Severity	Status
LOW	RESOLVED

The addresses of the factory, WETH, USDC, and Cookie contracts are checked to see if they are empty (equal to address(0)). These checks are unnecessary because the Quoter contract is created in the constructor of the Swapper contract, where these checks are already done.

RECOMMENDATION

Remove the unnecessary checks.

COOKIE3

audit / code review report

June 04, 2024

Severity	Status
LOW	RESOLVED

#8 - USE DIRECTLY ENCODED LEAVE INSTEAD OF HASHED PROOFS

The mapping proofUsed uses hashed proofs as key values. Instead of calling the hashBytes32Array function to encode and hash the given proofs, you can use the encoded leaf encodeLeaf(msg.sender, _amount) as a unique value.

The similar approach is used in Blur and LooksRare airdrop contracts.

RECOMMENDATION

Use directly encoded leave instead of hashed proofs.

contact@auditmos.com

COOKIE3

audit / code review report

June 04, 2024

#9 - THE SWAP CONTRACT UNNECESSARILY INHERITS THE OWNABLE CONTRACT

Severity	Status
LOW	RESOLVED

The Swap contract unnecessarily inherits the Ownable contract. The functionality of the Ownable contract is never used within the Swap contract and the owner does not have privileged rights to change anything within it. Inheriting the Ownable contract lacks purpose because its functionality is never utilized.

RECOMMENDATION

Inheriting of the Ownable contract should be removed.

contact@auditmos.com

auditmos.com

Secure your space

contact@auditmos.com