DIMAI

Smart Contract Audit Report

DimAI

June 2023

Disclaimer

This disclaimer is to inform you that the report you are reading has been prepared by Mantisec Labs for informational purposes only and should not be considered as investment advice. It is important to conduct your own independent investigation before making any decisions based on the information contained in the report. The report is provided "as is" without any warranties or conditions of any kind and Mantisec Labs excludes all representations, warranties, conditions, and other terms. Additionally, Mantisec Labs assumes no liability or responsibility for any kind of loss or damage that may result from the use of this report.

It is important to note that the analysis in the report is limited to the security of the smart contracts only and no applications or operations were reviewed. The report contains proprietary information, and Mantisec Labs holds the copyright to the text, images, photographs, and other content. If you choose to share or use any part of the report, you must provide a direct link to the original document and credit Mantisec Labs as the author.

By reading this report, you are accepting the terms of this disclaimer. If you do not agree with these terms, it is advisable to discontinue reading the report and delete any copies in your possession.

Introduction

About DimAI

As a portal into the metaverse creation, DimAI provides a one-stop NFT minting platform. After choosing AI creation or custom works, only a small amount of Meer is needed as the on-chain processing fee to easily generate NFTs. DimAI's powerful functions are applicable to all types of NFT distribution and support selling, trading or collecting on the trading platform. It effectively helps Web3 users to join the

NFT creation market, seize the new era windfall, and jointly meet the tide of metaverse innovation and application to virtualize and digitize the real world.

Audit Process & Methodology

Mantisec Labs team has performed thorough testing of the project, starting with analyzing the code design patterns in which we reviewed the smart contracts architecture to ensure it is structured and safe use of third-party smart contracts and libraries.
Our team then performed a formal line-by-line inspection of the Smart Contract to find potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.
In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors, including -

1. Structural analysis of the smart contract is checked and verified.
2. An extensive automated testing of all the contracts under scope is conducted.
3. Line-by-line Manual Code review is conducted to evaluate, analyze and identify the potential security risks in the contract.
4. Evaluation of the contract's intended behavior and the documentation shared is an imperative step to verify the contract behaves as expected.
5. For complex and heavy contracts, adequate integration testing is conducted to ensure that contracts perform acceptably while interacting.
6. Storage layout verifications in the upgradeable contract are a must.
7. An important step in the audit procedure is highlighting and recommending better gas optimization techniques in the contract.

Audit Goals

Mantisec Labs was commissioned by the DimAI team to perform an audit of their smart contracts. The audit was conducted in the month of **June 2023.**

The purpose of this audit was to achieve the following:

i. Identify potential security issues within the smart contract

ii. Formally check the logic behind the given smart contract

Information in this report should be used for understanding the risk exposure of this smart contract, and as a guide to improving the security posture by remediating the issues that were identified.

Audit Details

- Project Name: DimAI
- Contracts Name: DimAILogic.sol, DimaiNFT.sol, DimaiToken.sol
- Languages: Solidity(Smart contract)
- Platforms and Tools: Remix IDE, Solhint, VScode, Contract Library, Slither

Security Level Reference

Every issue in this report were assigned a severity level from the following:

High severity issues will bring problems and should be fixed.
Medium severity issues could potentially bring problems and should eventually be fixed.
Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future

Issues	High	Medium	Low
Open	0	0	0
Closed	2	0	2

Contract Name: DimAILogic.sol

High Severity issues

1. Incorrect Fee Calculation in useDimaiByMeer Function

Description:
There is an issue with the useDimaiByMeer function,In the DimAILogic contract.Which allows users to use the Dimai service by providing a payment and other parameters.Although, the function does not take into consideration for the case when the times parameter is set to zero. This omission leads to an incorrect fee calculation and potentially results in unrequired behavior.

Impact:
The lack of handling for the times parameter being zero can have the following impacts:

Incorrect Fee Calculation: When times is set to zero, the totalFee calculation becomes zero. This means that no fee is charged to the user, regardless of the payment provided. Consequently, the fee calculation is inaccurate and does not reflect the intended charging mechanism.

Payment Validation Bypass: Since the require statement require(msg.value >= totalFee, "Incorrect fee amount"); always evaluates to true when totalFee is zero, users can submit payments lower than the intended fee or even no payment at all. This bypasses the expected payment validation, potentially allowing unauthorized or free usage of the Dimai service.

Recommendation:

Validate the times parameter: Add a validation check at the beginning of the function to ensure that times is greater than zero. If times is zero, revert the transaction with an appropriate error message indicating that the fee multiplier must be greater than zero.

Medium Severity issues

Issues No issues were found.

Low Severity Issues

1. Reentrant Guard

Description:

The **buyDIM** and **swapDIM** functions do not explicitly protect against reentrancy attacks, which leaves it potentially vulnerable to such attacks. Reentrancy attacks occur when an external contract is called during the execution of a function, allowing the attacker to reenter the function before it completes and potentially manipulate the contract's state.

Potential Exploitation Scenario:

An attacker deploys a malicious contract that implements a function with a fallback or receive function.

The attacker initiates a transaction to the **buyDIM** or **swapDIM** functions from their malicious contract.

During the execution of the **buyDIM** or **swapDIM** function, the malicious contract's fallback or receive function is invoked, allowing the attacker to reenter the **buyDIM** or **swapDIM** function while it is still in progress.

The attacker can perform undesired actions, such as calling other functions in the contract, manipulating data, or even draining the contract's Ether balance.

Recommended Mitigation:

To mitigate the potential reentrancy vulnerability, Implement a reentrancy guard modifier to prevent reentrant calls. This can be achieved by setting a boolean variable to indicate whether the function is currently being executed and reverting if an attempt is made to reenter the function before completion.

2. Ambiguity in Event Emission for BuyDIM Transactions

Description:
In the DimAILogic contract, there is an ambiguity in the event emission related to the buyDIM transactions. The contract includes two functions, buyDIM and swapDIM, which enable users to purchase DIM tokens. However, both functions emit the same event, BuyDIM, without providing clear distinctions between the types of transactions. This lack of clarity can lead to confusion, making it difficult to differentiate between direct purchases and those executed based on a specific ratio. Consequently, this ambiguity hampers the accurate tracking, analysis, and interpretation of DIM token purchases within the contract.

Impact:
The confusion arising from the identical event emission can have several negative consequences:

Ambiguity in Event Logs: Emitting the same event for different types of transactions can make it challenging to differentiate between transactions executed via the buyDIM function and those executed via the swapDIM function. This ambiguity can hinder the ability to accurately analyze and interpret event logs.
User Experience: From a user's perspective, it may be confusing to receive event notifications that do not clearly indicate the type of transaction they initiated. This lack of clarity may lead to frustration and a suboptimal user experience.

Recommendation:
To address the confusion arising from the event emission, We recommend creating a new event specifically for the swapDIM function. This will provide a clear distinction between the two types of transactions and enhance the clarity and understanding of the emitted events.

Contract Name: DimaiNFT.sol

High Severity Issues

1. Lack of Access Control in the **freeMint** Function

Issue Description:
In the **DimaiNFT** contract, the **freeMint** function lacks proper access control, allowing anyone to call the function and mint tokens without any associated cost. This absence of access control poses a security risk and can lead to unauthorized minting of tokens.

Impact:
The lack of access control in the **freeMint** function can have various negative impacts:

Unauthorized Token Minting: Any user, including potential attackers, can freely call the **freeMint** function and mint tokens without incurring any cost. This unauthorized minting can lead to the creation of an excessive number of tokens or tokens being minted by malicious actors, potentially devaluing the token economy or causing disruption to the intended system.

Misuse of Resources: The unrestricted access to mint tokens can lead to the misuse of computational resources, as attackers or unintended users can overload the system by repeatedly calling the freeMint function and minting a large number of tokens without incurring any associated cost.

Recommendation:
Apply an access control modifier, such as **onlyOwner**, to the **freeMint** function. This ensures that only the contract owner or a designated authority can call the function and mint tokens. The onlyOwner modifier restricts access to trusted addresses and provides an additional layer of security.

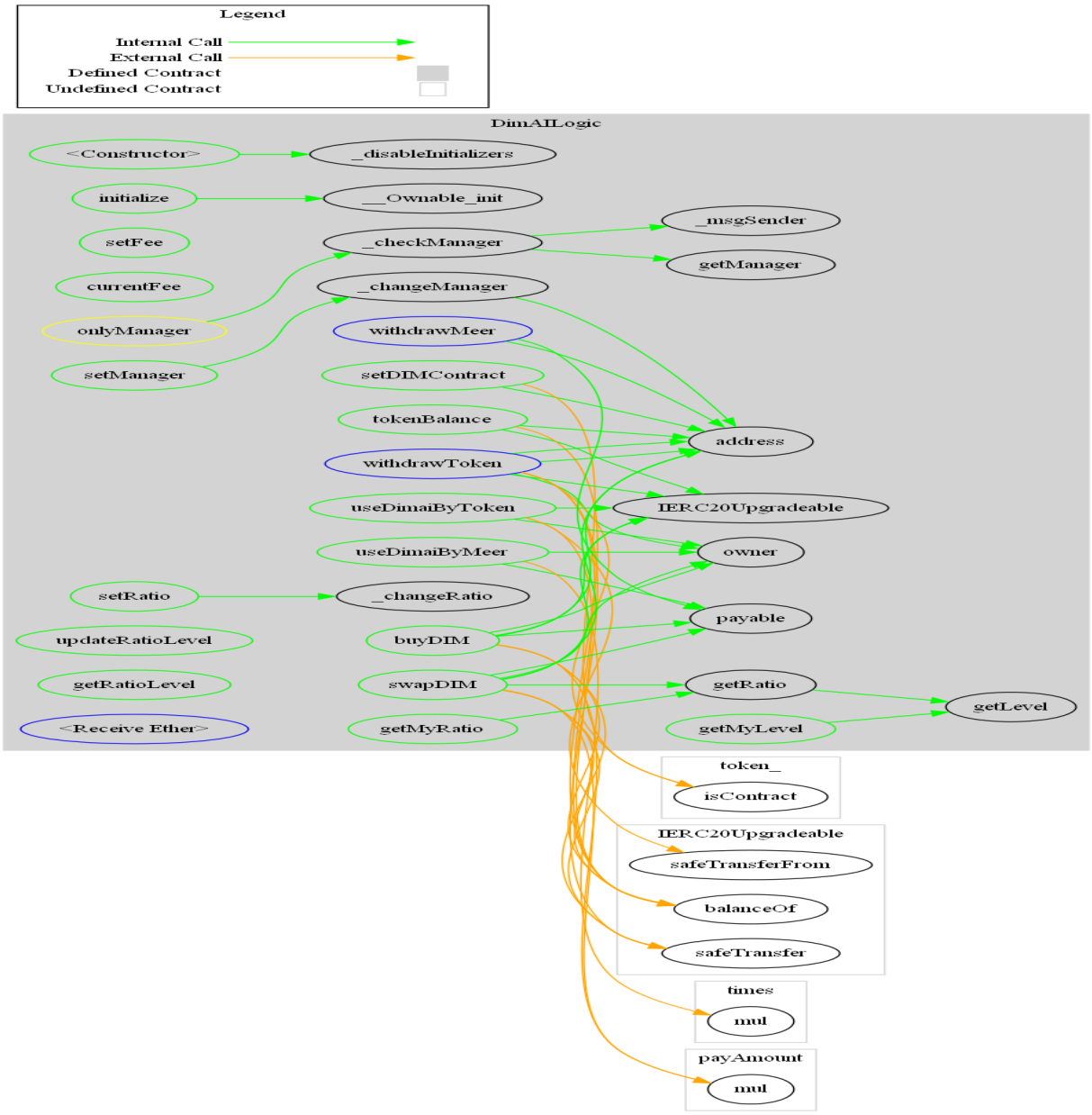
Medium Severity Issues
Issues No issues were found.
Low Severity Issues
Issues No issues were found.

Contract Name: DimaiToken.sol

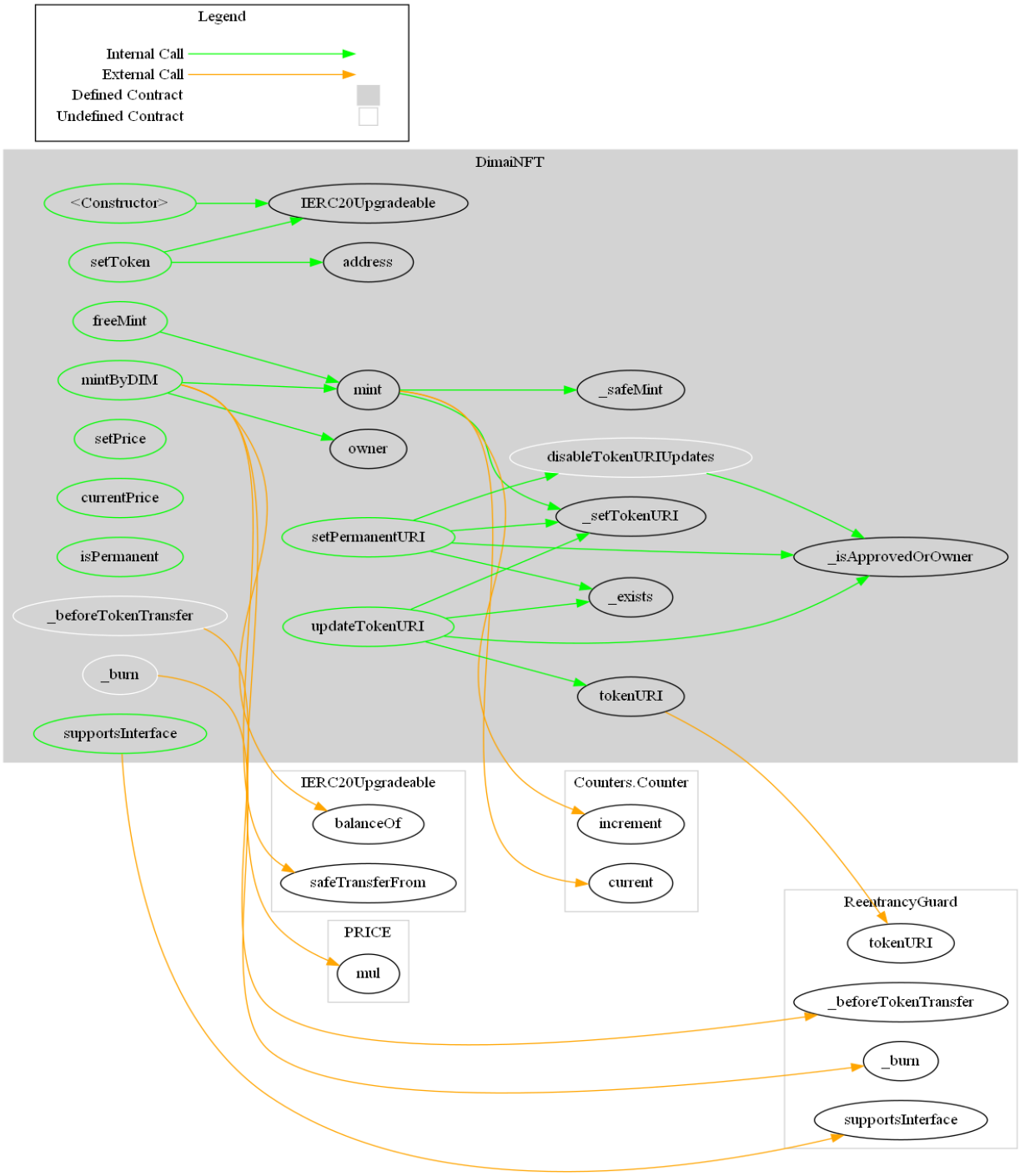
High Severity Issues
Issues No issues were found.
Medium Severity Issues
Issues No issues were found.
Low Severity Issues
Issues No issues were found.

Additional Details

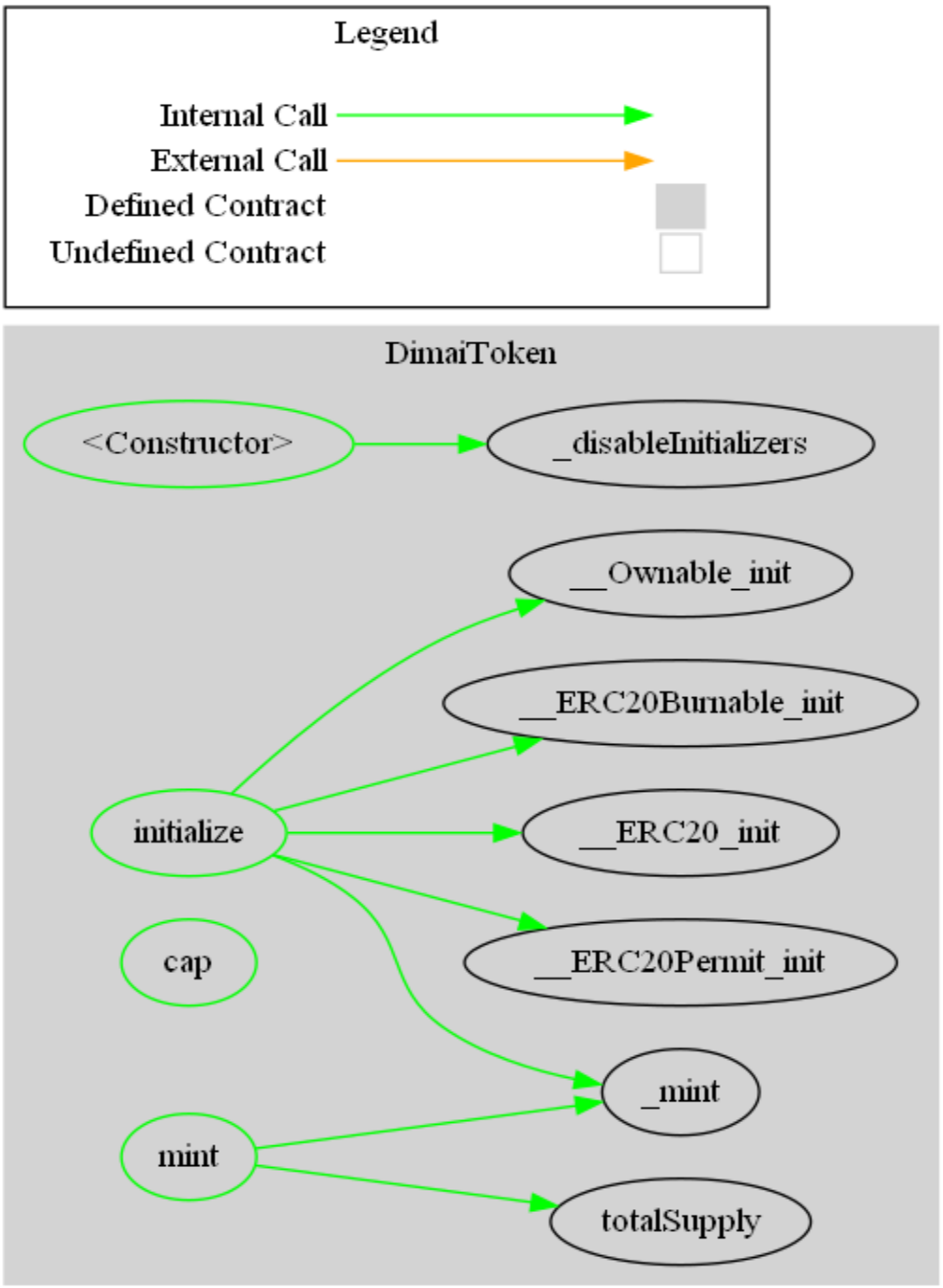
1. **DimAILogic.sol**


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
DimAILogic
<Constructor>
_disableInitializers
initialize
__Ownable_init
setFee
_checkManager
_msgSender
getManager
currentFee
_changeManager
onlyManager
withdrawMeer
setManager
setDIMContract
tokenBalance
address
withdrawToken
useDimaiByToken
IERC20Upgradeable
useDimaiByMeer
owner
setRatio
_changeRatio
payable
updateRatioLevel
buyDIM
getRatioLevel
swapDIM
getRatio
getLevel
<Receive Ether>
getMyRatio
getMyLevel
token_
isContract
IERC20Upgradeable
safeTransferFrom
balanceOf
safeTransfer
times
mul
payAmount
mul

2. DimaiNFT



3. DimaiToken


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
DimaiToken
<Constructor>
_disableInitializers
__Ownable_init
__ERC20Burnable_init
initialize
__ERC20_init
cap
__ERC20Permit_init
_mint
mint
totalSupply

Concluding Remarks

While conducting the audit of the DIMAI smart contracts, it was observed that the contracts contained High and Low severity issues.
Our auditors confirmed that High and Low severity issues are now resolved by the developers.