0x Guard

Smart contracts security assessment

Final report

[Tariff: Standard]

Defender Finance Genesis

December 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Defender Finance Genesis**.

The Genesis contract allows users to farm tokens in different pools.

The code is available at the GitHub repository and was audited after the commit 0ad7d697e30ea16daeed3e7fb77acc97744bf05f.

The inspected contract is Genesis.sol.

Report Update.

The contract's code was updated according to this report and rechecked after the commit 5380d63189b3aeb523681387b895e6d6c717cfff.

Name	Defender Finance Genesis
Audit date	2022-12-08 - 2022-12-08
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
Genesis	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) of all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed

Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

No issues were found

Low severity issues

1. Variable default visibility (Genesis)

Status: Fixed

The variables `lastPolRewardTime`, `lastDevRewardTime` have default visibility. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

2. Gas optimization (Genesis)

Status: Fixed

The variables `operator`, `rewardTokenPerSecondForUser`, `rewardTokenPerSecondForPol`, `rewardTokenPerSecondForDev`, `devWallet`, `polWallet` can be declared as `immutable` to save gas.

Conclusion

Defender Finance Genesis Genesis contract was audited. 2 low severity issues were found.
2 low severity issues have been fixed in the update.

In the updated code, the deposit fee has been removed from the contract code.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard 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 considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.
OxGuard retains exclusive publishing rights for the results of this audit on its website and social networks.

Slither output

```
Genesis.pending(uint256,address) (contracts/Genesis.sol#169-185) performs a
multiplication on the result of a division:
        - _rewardTokenReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/Genesis.sol#176)
        - accRewardTokenPerShare =
accRewardTokenPerShare.add(_rewardTokenReward.mul(1e18).div(tokenSupply)) (contracts/
Genesis.sol#177)
Genesis.updatePool(uint256) (contracts/Genesis.sol#230-250) performs a multiplication
on the result of a division:
        - _rewardTokenReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/Genesis.sol#246)
        - pool.accRewardTokenPerShare =
pool.accRewardTokenPerShare.add(_rewardTokenReward.mul(1e18).div(tokenSupply))
(contracts/Genesis.sol#247)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

Genesis.updatePool(uint256) (contracts/Genesis.sol#230-250) uses a dangerous strict
equality:
        - tokenSupply == 0 (contracts/Genesis.sol#236)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Genesis.updatePool(uint256) (contracts/Genesis.sol#230-250) has costly operations
inside a loop:
        - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
Genesis.sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

Pragma version0.8.13 (contracts/Genesis.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
```

0x Guard