0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

The Story of Draco

June 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for The Story of Draco.

Name	The Story of Draco
Audit date	2022-06-07 - 2022-06-07
Language	Solidity
Platform	Fantom Network

Contracts checked

Name	Address
ERC20	0x01D3569eEdD1Dd32A698CAB22386d0F110d6b548
ControllerChef	0xAedCc6E2710d2E47b1477A890C6D18f7943C0794
Draco	0x01D3569eEdD1Dd32A698CAB22386d0F110d6b548
Disaster	0xb1AF73E4541A24894858fB6Ad9eE872e92caAEe5

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) 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	failed
Outdated Compiler Version	passed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. Gas optimisation (ERC20)

a. `_name`, `_symbol` should me marked as immutable;

b. `MAXSUPPLY` should be const.

2. Gas optimisation (ControllerChef)

Direct Boolean comparison L973. Can be replaced with negation symbol:

```
require(!poolExistence[_lpToken], "nonDuplicated: duplicated");
```

3. Optional massUpdatePools() (ControllerChef)

In `set()` and `add()` functions mass pools update is optional. This may cause unfair distribution for rarely updated pools with few stakeholders. If such pools are not updated before new allocation is set they will get fewer rewards than they should, since rewards since `lastRewardperBlock` will be calculated according to modified allocation where they have a smaller part.

4. Pending rewards miscalculation (ControllerChef)

The mistake in rewards estimation is caused by gross violation of basic fraction addition rule.

```
uint256 distribution = dracoReward / 110;
```

L1067 `distribution` variable stores the sum of additional mints to `treasuryAdd` and `disasterAdd` equal to 1/100 and 1/10 of the `dracoReward` respectively. To find the sum of fractions, it's needed to bring them to a common denominator and then sum their numerators, so the the sum of 1/100 and 1/10 is 11/100. Whereas now it's simple denominators sum. Correct distribution calculations would be following:

```
uint256 distribution = dracoReward * 11 / 100;
```

Conclusion

The Story of Draco ERC20, ControllerChef, Draco, Disaster contracts were audited. 4 low severity issues were found.

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.

x Guard