



d3ploy



SECURITY ASSESSMENT

Param Labs

November 24th 2023

TABLE OF Contents

01 Legal Disclaimer

02 D3ploy Intro

03 Project Summary

04 Audit Score

05 Audit Scope

06 Methodology

07 Key Findings

08 Vulnerabilities

09 Source Code

10 Appendix

LEGAL

Disclaimer

D3ploy audits are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are 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 d3ploy to perform a security review. D3ploy 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.

D3ploy audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort. The report is provided only for the contract(s) mentioned in the report and does not include any other potential additions and/or contracts deployed by Owner. The report does not provide a review for contract(s), applications and/or operations, that are out of this report scope.

D3ploy’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.

D3ploy represents an extensive auditing 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. D3ploy’s position is that each company and individual are responsible for their own due diligence and continuous security. The security audit is not meant to replace functional testing done before a software release. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits and a public bug bounty program to ensure the security of the smart contracts.

D3PLOY

Introduction

D3ploy is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

Secure your project with d3ploy

We offer field-proven audits with in-depth reporting and a range of suggestions to improve and avoid contract vulnerabilities. Industry-leading comprehensive and transparent smart contract auditing on all public and private blockchains.



Vulnerability checking

A crucial manual inspection carried out to eliminate any code flaws and security loopholes. This is vital to avoid vulnerabilities and exposures incurring costly errors at a later stage.



Contract verification

A thorough and comprehensive review in order to verify the safety of a smart contract and ensure it is ready for launch and built to protect the end-user



Risk assessment

Analyse the architecture of the blockchain system to evaluate, assess and eliminate probable security breaches. This includes a full assessment of risk and a list of expert suggestions.



In-depth reporting

A truly custom exhaustive report that is transparent and depicts details of any identified threats and vulnerabilities and classifies those by severity.



Fast turnaround

We know that your time is valuable and therefore provide you with the fastest turnaround times in the industry to ensure that both your project and community are at ease.

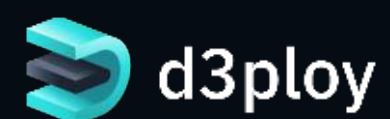


Best-of-class blockchain engineers

Our engineers combine both experience and knowledge stemming from a large pool of developers at our disposal. We work with some of the brightest minds that have audited countless smart contracts over the last 4 years.



WEBSITE **d3ploy.co**



@d3ploy_ TWITTER

PROJECT

Introduction

Param Labs is an independent game and technology development studio, specializing in multiplayer blockchain games, AAA design, and innovative technology development.

They are focused on leveraging emerging technology to provide user generated value, digital-ownership of gaming assets, and unique gaming experiences to the mass market while seamlessly connecting the worlds of web2 and web3.

Project Name *Param Labs*

Contract Name *PARAM Token*

Contract Address *Not Yet Deployed on Mainnet*

Contract Chain -

Contract Type *Smart Contract*

Platform *EVM*

Language *Solidity*

Network *ERC20*

Codebase *Private GitHub Repository*

Total Supply -

INFO

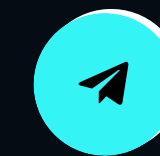
Social



<https://paramlabs.io/>



<https://twitter.com/ParamLaboratory>



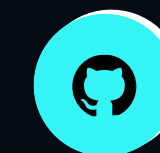
-



-



-



-



-

95

PASS

AUDIT

Score

✦ Issues	3
✦ Critical	0
✦ Major	0
✦ Medium	0
✦ Minor	2
✦ Informational	1
✦ Discussion	0

All issues are described in further detail on the following pages.

AUDIT Scope

GITHUB REPOSITORY

/ez7212/ParamToken/contracts

LOCATION

✦ Private GitHub Repository

WEBSITE

d3ploy.co



d3ploy

@d3ploy_

TWITTER

REVIEW Methodology

TECHNIQUES

This report has been prepared for Param Labs to discover issues and vulnerabilities in the source code of the Param Labs project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic, Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective in the comments below.

TIMESTAMP

Version *v1.0*

Date *2023/11/23*

Description *Layout project*

Architecture / Manual review / Static & dynamic security testing

Summary

Version *v1.1*

Date *2023/11/24*

Description *Re-audit addressed issues*

Final Summary

KEY Finding

TITLE	SEVERITY	STATUS
Floating and Outdated Pragma	✦ Minor	Acknowledged
Approve Frontrunning Attack	✦ Minor	Acknowledged
Absence of Token Burn Function	✦ Informational	Acknowledged

IN - DEPTH Vulnerabilities

1

DESCRIPTION

Locking the pragma helps ensure that the contracts do not accidentally get deployed using an older version of the Solidity compiler affected by vulnerabilities. The contract was allowing floating or unlocked pragma to be used, i.e., ^0.8.0. This allows the contracts to be compiled with all the solidity compiler versions above the limit specified.

VULNERABLE CODE

- ParamToken.sol – pragma solidity ^0.8.0

IMPACTS

If the smart contract gets compiled and deployed with an older or too recent version of the solidity compiler, there's a chance that it may get compromised due to the bugs present in the older versions or unidentified exploits in the new versions. Incompatibility issues may also arise if the contract code does not support features in other compiler versions, therefore, breaking the logic. The likelihood of exploitation is really low therefore this is only informational.

Issue : Floating and Outdated Pragma

Level : Minor

Type : Floating Pragma

Remediation : Keep the compiler versions consistent in all the smart contract files. Do not allow floating pragmas anywhere. It is suggested to use the 0.8.22 pragma version

Reference: <https://swcregistry.io/docs/SWC-103>

Alleviation / Retest : ParamLabs team acknowledged the issue.

DESCRIPTION

Approve is well known to be vulnerable to front-running attacks. This may be exploited in cases where in case the user decides to modify the spending amount in quick succession, and the attacker sees the change and transfers more tokens than required. The exploitation involves two parties - a victim and an attacker and the attacker must be monitoring the transactions to front-run the transaction.

VULNERABLE CODE

- ParamToken.sol - function approve()

IMPACTS

Front-running attacks might allow attackers to withdraw/transfer more tokens than the victim initially intended to allow.

Issue : Approve Frontrunning Attack

Level : Minor

Type : Frontrunning

Remediation : Instead of using approve() to change the allowance, it is recommended to use safeIncreaseAllowance and safeDecreaseAllowance functions which are meant for this use case.

References:

- https://docs.google.com/document/d/1YLPtQxZuIUAvO9cZIO2RPXBbT0mooh4D_YKjA_jp-RLM/edit
- [https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts / token/ERC20/ERC20.sol](https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol)
- <https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts / token/ERC20/Utils/SafeERC20.sol>

Alleviation / Retest : ParamLabs team acknowledged the issue.

DESCRIPTION

The provided Token contract does not include a function to burn tokens. The absence of a token burn mechanism limits the contract's flexibility and functionality, as it does not allow for the removal of tokens from circulation. A burn function is essential for various purposes, including reducing the total supply, implementing deflationary mechanisms, or complying with specific tokenomics

VULNERABLE CODE

- ParamToken.sol

Issue : Absence of Token Burn Function

Level : Informational

Type : Best Practices

Remediation : It is recommended to implement a token burn function in the contract with proper access controls.

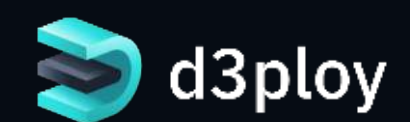
Alleviation / Retest : ParamLabs team acknowledged the issue.

SOURCE Code

GitHub Repository

/ez7212/ParamToken/contracts

WEBSITE **d3ploy.co**



d3ploy

@d3ploy_ *TWITTER*

REPORT Appendix

FINDING CATEGORIES

The assessment process will utilize a mixture of static analysis, dynamic analysis, in-depth manual review and/or other security techniques.

This report has been prepared for Param Labs project using the above techniques to examine and discover vulnerabilities and safe coding practices in Param Labs's smart contract including the libraries used by the contract that are not officially recognized.

A comprehensive static and dynamic analysis has been performed on the solidity code in order to find vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds.

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The testing methods find and flag issues related to gas optimizations that help in reducing the overall gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

AUDIT SCORES

D3ploy Audit Score is not a live dynamic score. It is a fixed value determined at the time of the report issuance date.

D3ploy Audit Score is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports and scores are 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 d3ploy to perform a security review.



d3ploy

WEBSITE d3ploy.co @d3ploy_ TWITTER