d3ploy

d3ploy

SECURITY ASSESSMENT

Vision Game

December 26th 2022

TABLE OF

Contents

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.

Vunerability 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.

PROJECT

Introduction

VisionGame brings the traditional game publishing experience, boosted for the blockchain. A suite of unique products, technical and creative services, all to support the ever-growing gaming blockchain industry, raising the bar one game at a time.

At the core of Vision's technical solution, stands VisionSDK. Through a mix of centralized and decentralized architecture, our SDK allows developers to keep the focus on their craft while we take care of everything else.

Project Name Vision Game
Contract Name VISION Token
Contract Address HVkFqcMHevVPb4XKrf4XowjEaVVsBoqJ2U1EG59Dfk5j (Solana)
Contract Chain Mainnet
Contract Type Smart Contract
Platform EVM
Language Solidity
Network BNB Chain (BEP20) + Solana (SOL)
Codebase Private GitHub Repository
Total Token Supply 1,000.000.000

INFO

Social

https://visiongame.io/

https://twitter.com/visiongame_

https://t.me/visiongame_ann

-

https://medium.com/@visiongame

-

cristian@visiongame.studio

95

PASS

AUDIT
Score

✦ Issues	4
✦ Critical	0
✦ Major	0
✦ Medium	0
✦ Minor	2
✦ Informational	2
✦ Discussion	0

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

AUDIT Scope

CODEBASE FILES	LOCATION
VISION.sol	✦ contracts/VISION.sol
tVISION.sol	✦ contracts/tVISION.sol
Faucet.sol	✦ contracts/Faucet.sol

REVIEW Methodology

TECHNIQUES

This report has been prepared for Vision Game to discover issues and vulnerabilities in the source code of the Vision Game 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	*2022/12/22*
Descrption	*Layout project* *Architecture / Manual review / Static & dynamic security testing* *Summary*

Version	*v1.1*
Date	*2022/12/26*
Descrption	*Reaudit addressed issues* *Final Summary*

KEY Finding

TITLE	SEVERITY	STATUS
Floating Pragma	✦ Minor	Fixed
Large Number Literals	✦ Gas	Fixed
Missing Events In Important Functions	✦ Minor	Acknowledged
Functions Should Be Declared External	✦ Gas	Fixed

IN - DEPTH Vulnerabilities

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 contracts found in the repository were allowing floating or unlocked pragma to be used, i.e., ^0.8.9.
This allows the contracts to be compiled with all the solidity compiler versions above 0.8.9.

AFFECTED CODE

- VISION.sol
- tVISION.sol
- Faucet.sol

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 Pragma

Type : Floating Pragma (SWC-103)

Level : Low

Remediation : Keep the compiler versions consistent in all the smart contract files. Do not allow floating pragmas anywhere.

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

Alleviation / Retest : The team opted to consider our references and applied the recommendation.

IN - DEPTH Vulnerabilities

DESCRIPTION

Solidity supports multiple rational and integer literals, including decimal fractions and scientific notations. The use of very large numbers with too many digits was detected in the code that could have been optimized using a different notation also supported by Solidity.

AFFECTED CODE

- VISION.sol L19

IMPACTS

Having a large number literals in the code increases the gas usage of the contract while its deployment and when the functions are used or called from the contract. It also makes the code harder to read and audit and increases the chances of introducing code errors.

Issue : Large Number Literals

Type : Gas & Missing Best Practices

Level : Gas

Remediation : This can be rewritten as 1e27. Scientific notation in the form of 2e10 is also supported, where the mantissa can be fractional but the exponent has to be an integer. The literal MeE is equivalent to M * 10**E. Examples include 2e10, 2e10, 2e-10, 2.5e1, as suggested in official solidity documentation. https://docs.soliditylang.org/en/latest/types.html#rational-and-integer-literals

Alleviation / Retest : The VisionGame team heeded our references and applied the suggested recommendation.

IN - DEPTH Vulnerabilities

DESCRIPTION

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain. These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.

The contract was found to be missing these events on certain critical functions which would make it difficult or impossible to track these transactions off-chain.

AFFECTED CODE

The following functions were affected

- Faucet.sol - setTokenAmountPerDrip() - L34
  setWaitTimeBetweenDrips() - L38

IMPACTS

Events are used to track the transactions off-chain and missing these events on critical functions makes it difficult to audit these logs if they're needed at a later stage.

Issue : Missing Events in Important Functions

Type : Missing Best Practices

Level : Low

Remediation : Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Alleviation / Retest : The VisionGame team commented that this is only going to be for testnet so it is irrelevant.

IN - DEPTH Vulnerabilities

DESCRIPTION

Public functions that are never called by a contract should be declared external in order to conserve gas. The following functions were declared as public but were not called anywhere in the contract, making public visibility useless.

AFFECTED CODE

- Faucet.sol - pause() - L42
  unpause() - L46
  dripTokensTo() - L27
  setTokenAmountPerDrip() - L34
  setWaitTimeBetweenDrips() - L38
- VISION.sol - snapshot() - L22
  pause() - L26
  unpause() - L30

IMPACTS

Smart Contracts are required to have effective Gas usage as they cost real money, and each function should be monitored for the amount of gas it costs to make it gas efficient. Public functions cost more Gas than external functions.

Issue : Functions Should Be Declared External

Type : Gas Optimization

Level : Gas

Remediation : Use the external state visibility for functions that are never called from inside the contract.

Alleviation / Retest : The team opted to consider our references and applied the recommendation.

SOURCE Code

Raw Solidity Files

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 Vision Game project using the above techniques to examine and discover vulnerabilities and safe coding practices in Vision Game'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