DeHacker

Code Security Assessment

Vita Inu

July 2 6 th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

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

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Vita Inu
Platform	BSC
Website	https://vitainu.org/
Type	meme
Language	Solidity

Vulnerability Summary

Vulnerability Level	**Total**	Mitigated	**Declined**	**Acknowledged**	**Partially Resolved**	**Resolved**
Critical	0	0	0	0	0	0
Major	2	1	0	1	0	0
Medium	0	0	0	0	0	0
Minor	1	0	0	1	0	0
Informational	2	0	0	2	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
BEP	BEP20Vinu.sol	da8173b8f3fe54a5057530120cb63a8e9c3bf12ada1d39243309b7c8d846fe42

Findings

ID	Category	Severity	Status
BEP-01	Centralization / Privilege	Major	Mitigated
BEP-02	Centralization / Privilege	Major	Acknowledged
BEP-03	Gas Optimization	Minor	Acknowledged
BEP-04	Volatile Code	Informational	Acknowledged
BEP-05	Gas Optimization	Informational	Acknowledged

MAJOR

BEP-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization/ Privilege	Major	BEP20Vinu.sol: 357	Mitigated

Description

All of the Vinu tokens are sent to the contract deployer when deploying the contract. This could be acentralization risk as the deployer can distribute Vinu tokens without obtaining the consensus of thecommunity.Up to May.5, on-chain information indicates that 89% of the total supply of Vinu tokens are still withincontract owner's wallet.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the teamshall make enough efforts to restrict the access of the private key.

MAJOR

BEP-02 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	BEP20Vinu.sol : 317, 326, 498	Acknowledged

Description

In the contract BEP20Vinu the role owner has authority over the following functions:

mint()
renounceO
wnership()
transferOwnership()

Any compromise to the owner account may allow a hacker to take advantage of this authority and takecontrol over token minting.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the securityoperation and level of decentralization, which in most cases cannot be resolved entirely at the presentstage. We advise the client to carefully manage the privileged account's private key to avoid any potentialrisks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol beimproved via a decentralized mechanism or smart-contract-based accounts with enhanced securitypractices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a differentlevel in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (,) combination mitigate by delaying the sensitive operation and avoiding asingle point of key management failure.

Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to theprivate key compromised;
AND
A medium/blog link for sharing the timelock contract and multi-signers addresses information withthe public audience.

Long Term:
Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Introduction of a DAO/governance/voting module to increase transparency and user involvement;
AND
A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO informationwith the public audience.

Permanent:
Renouncing the ownership or removing the function can be considered fully resolved.

Renounce the ownership and never claim back the privileged roles;
OR
Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shallmake a decision based on the current state of their project, timeline, and project resources.

MINOR

BEP-03 | Variable Declare As Immutable

Category	Severity	Location	Status
Gas Optimization	Minor	BEP20Vinu.sol: 348~350	Acknowledged

Description

The variables _name , _symbol , and _decimals assigned in the constructor can declare with Immutable .Immutable state variables can be assigned during contract creation but will remain constant throughoutthe lifetime of a deployed contract. A big advantage of immutable variables is that reading them issignificantly cheaper than reading from regular state variables since will not be stored in storage. Still,values will directly insert the values into the runtime code.

Recommendation

We recommend using an immutable state variable for the mentioned state variables.

INFORMATIONAL

BEP-04 | Redundant Statements

Category	Severity	Location	Status
Volatile Code	Informational	BEP20Vinu.sol: 597~600	Acknowledged

Description

Internal function _burnFrom() do not affect the functionality of the codebase and appear to be eitherleftovers from test code or older functionality.

Recommendation

We advise that they are removed to better prepare the code for production environments.

INFORMATIONAL

BEP-05 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	BEP20Vinu.sol : 466, 485, 498, 506	Acknowledged

Description

The following functions are declared as public , contain array function arguments, and are not invoked inany of the contracts contained within the project's scope. The functions that are never called internallywithin the contract should have external visibility.

increaseAllowance()
decreaseAllowance()
mint()
burn()

Recommendation

We advise that the functions' visibility specifiers are set to external to optimize the gas cost of thefunction.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

- Ethereum
- Cosmos
- Eos
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

July 2023