DeHacker

Code Security Assessment

DRIVE 2

June 16th, 2022

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	DRIVE 2
Platform	BSC
website	https://drive2.cc/
Type	GameFi
Language	Solidity

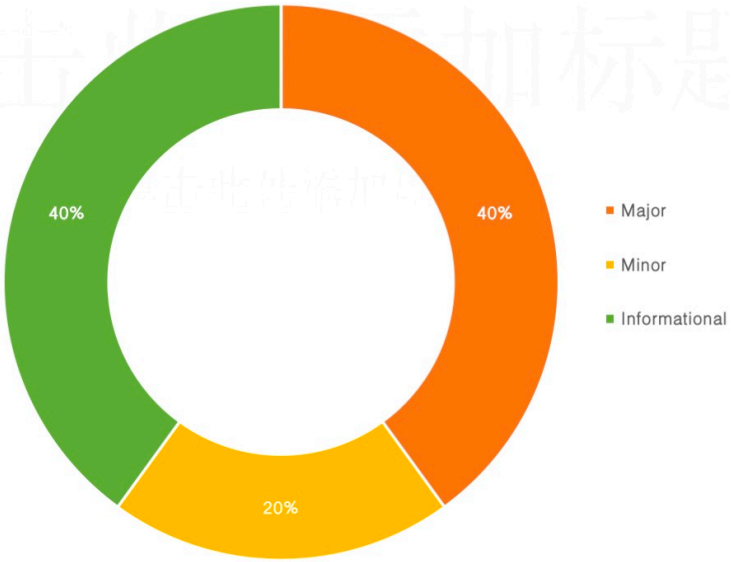
Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
VTT	vesting-token.sol	1787f52497bfc5776d751268387a0ee391c544826a6cf5e7a62eade3dbe20b25

Findings



ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	Centralization / Privilege	Major	Acknowledged
ADG-01	Incorrect Modifier Name	Coding Style	Informational	Resolved
ADG-02	Missing Emit Events	Coding Style	Informational	Resolved
CCV-01	Missing Param Check	Logical Issue	Minor	Resolved
GHK-01	Unnecessary AvailableBalance	Logical Issue	Major	Acknowledged

Major

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major		Acknowledged

Description

In the contract deployed, the role superAdmin has the authority over the following function:

- function createBatchVesting()
- function setBatchVersionTime()
- function createVesting()
- function setVersionTime()

Any compromise to the accounts may allow a hacker to take advantage of this authority and causes security problems.

Recommendation

We advise the client to carefully manage the OWNER account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multi-signature wallets.

Here are some feasible suggestions that would also mitigate this risk in the **short-term** and **long-term**:

- A time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The DRIVE 2 team informed us that the requirement has been met without modification.

GHK-01 | Unnecessary Available Balance

Category	Severity	Location	Status
Logical Issue	Major	vesting-token.sol: 396	Acknowledged

Description

Without availableBalance check, it's still safe. Usually, swaps and other DEFI contracts using approve(contractAddress, uint256.max) and transferFrom() for the token deposit, but with availableBalance check, approve(contractAddress, uint256.max) is going fail.

Recommendation

Remove availableBalance check.

Alleviation

The DRIVE 2 team informed us that the contract needs this function.

Minor

CCV-01 | Missing Param Check

Category	Severity	Location	Status
Logical Issue	Minor	vesting-token.sol: 116	Resolved

Description

If cliff is 0, the function `_nowReleaseCount()` will get an error.

Recommendation

Add require(_cliff > 0, "Cliff is 0").

Alleviation

The error has been properly fixed by DRIVE 2 team.

Informational

ADG-01 | Incorrect Modifier Name

Category	Severity	Location	Status
Coding Style	Informational	vesting-token.sol: 30	Resolved

Description

According to the require, the error info "Vesting already exists" means in this modifier, `_beneficiary` should have no vesting, but the modifier name is `haveVesting`.

Recommendation

We advise the name `haveVesting()` change to `haveNoVesting()`.

Alleviation

The DRIVE 2 team has changed the name `haveVesting()`to `haveNoVesting()`.

ADG-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	vesting-token.sol: 161	Resolved

Description

There's no event logs there, but `setVersionTime()` is important for vesting, emit events would be better.

Recommendation

We advise adding emit events in the function `setVersionTime()`.

Alleviation

The DRIVE 2 team has added emit events in the function `setVersionTime()`.

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/

June 2022