DeHacker

Code Security Assessment

Hooked Protocol

January 30th, 2025

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. 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	Hooked Protocol
Platform	BSC
Website	https://hooked.io
Type	GameFi
Language	Solidity
Codebase	https://github.com/DEVHooked/release

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	1	0	0	0	0	1
Informational	3	0	0	0	0	3
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
EDE	Ecosystem.sol	1826e472eb0716926ed172c62b8013c96d76b4d9ae2e0211449511317ec741d1
PSD	PrivateSale.sol	43f547c0a8dd88a02c5708a300655427574746a31f7868b90d5d47338465acf9

Findings

ID	Title	Severity	Status
PSD-01	Divide Before Multiply	Minor	Resolved
DEV-01	Third Party Dependency	Informational	Resolved
DEV-02	Redundant Parameter	Informational	Resolved
DEV-04	Redundant import Statement	Informational	Resolved

MINOR

PSD-01 | Divide Before Multiply

Issue	Severity	Location	Status
Mathematical Operations	Minor	PrivateSale.sol:100	Resolved

Description

Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

```
100 return totalAllocation / 10 + (totalAllocation * 9 / 10 *elapsedTime) /
_unlockTimestamps.length;
```

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

INFORMATIONAL

DEV-01 | Third Party Dependency

Issue	Severity	Location	Status
Volatile Code	Informational	Ecosystem.sol: 78; PrivateSale.sol: 76	Resolved

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treatsthird party entities as black boxes and assume their functional correctness. However, in the real world, third parties canpossibly create severe impacts, such as lock tokens, etc.

```
78 function releasedAmount(address token, uint64 timestamp) public
viewvirtual returns (uint256) {
```

- The function Ecosystem.releasedAmount interacts with third party contract with IERC20 interface via token .

```
76 function releasedAmount(address token, uint64 timestamp) public
viewvirtual returns (uint256) {
```

- The function PrivateSale.releasedAmount interacts with third party contract with IERC20 interface via token .

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantlymonitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

INFORMATIONAL

DEV-02 | Redundant Parameter

Issue	Severity	Location	Status
Coding Style	Informational	Ecosystem.sol: 78; PrivateSale.sol: 76	Resolved

Description

The releasable token addresses in contracts are stored in the immutable variable _token , making the parameter token in the releasedAmount() function redundant.

Recommendation

Consider removing parameter token from the releasedAmount() function and using the _token variable to get the balance.

INFORMATIONAL

DEV-04 | Redundant import Statement

Issue	Severity	Location	Status
Coding Style	Informational	Ecosystem.sol: 5; PrivateSale.sol: 5	Resolved

Description

The imported Address library is never used in PrivateSale and Ecosystem contracts.

Recommendation

We recommend removing redundant import statements.

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

January 30th 2025