DeHacker

Code Security Assessment

Beyond Protocol

Aug 15 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	Beyond Protocol
Platform	Ethereum
Website	https://beyond.link/
Type	DeFi
Language	Solidity

Vulnerability Summary

Vulnerability Level	**Total**	Mitigated	**Declined**	**Acknowledged**	**Partially Resolved**	**Resolved**
Critical	0	0	0	0	0	0
Major	1	0	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
ERC	ERC20.sol	8e1aa7d97ba8ed4743ae9f249fd75c59328 8c7479e34d2e2a6fee97a11f1631f

Findings

ID	Category	Severity	Status
ERC-01	Centralization / Privilege	Major	Acknowledged
ERC-02	Coding Style	Informational	Acknowledged
ERC-03	Volatile Code	Minor	Acknowledged
ERC-04	Gas Optimization	Informational	Acknowledged

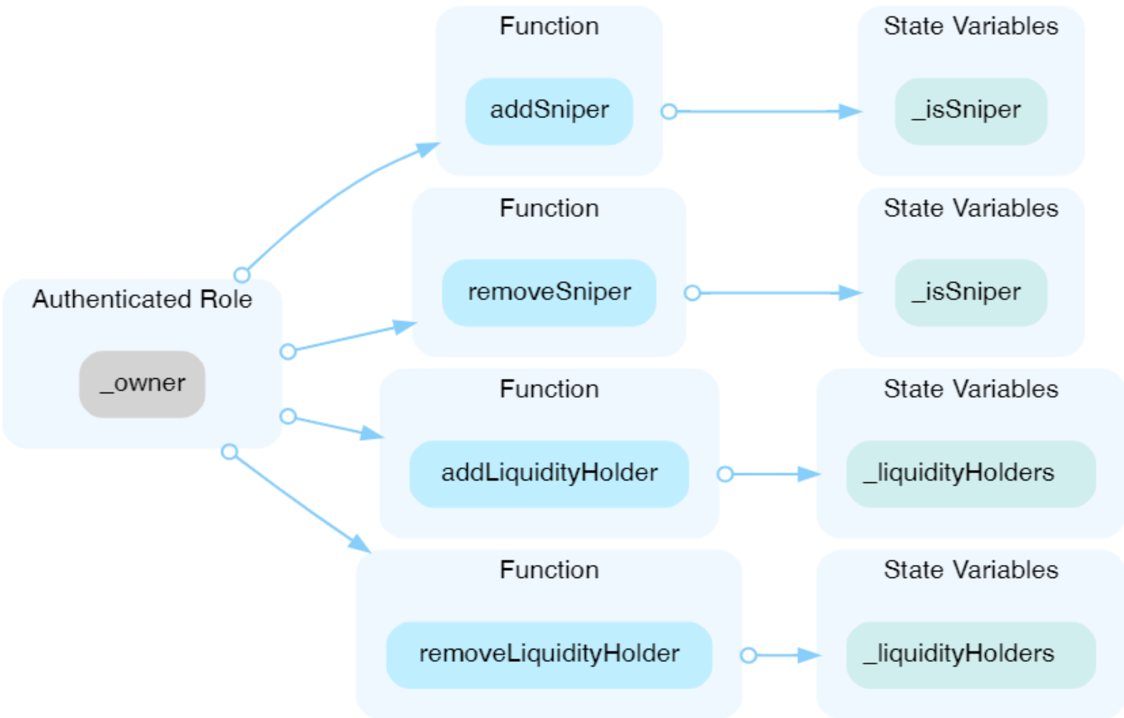
MAJOR

ERC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization /Privilege	Major	ERC20.sol: 1088~ 1093, 1095~1099, 1101~1103,1105~ 1107	Mitigated

Description

In the contract, ERC20 , the role, _owner , has the authority over the functions shown in the diagram below.Any compromise to the privileged account which has access to _owner may allow the hacker to takeadvantage of this.



ERC20.sol: 1088

Recommendation

We advise the client to carefully manage the privileged account's private key to avoid any potential risks ofbeing hacked.

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

Multisignature wallets.
Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the differentlevel in term of short-term and long-term:
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 theprivate key;
Introduction of a DAO/governance/voting module to increase transparency and user involvement.

INFORMATIONAL

ERC-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	ERC20.sol: 1088	Acknowledged

Description

The functions that affect the status of sensitive variables should be able to emit events as notifications.
addSniper()
removeSniper()
addLiquidityHolder()
removeLiquidityHolder()

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the function.

MINOR

ERC-03 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	ERC20.sol: 788	Acknowledged

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

```
788  constructor (string memory name_, string memory symbol_, address uniswapRouter,
uint256 totalMint, uint256 _snipeBlockAmt) {
789      require(uniswapRouter != address(0), "uniswapRouter is address 0");
790      _name = name_;
791      _symbol = symbol_;
792      ...
793  }
```

INFORMATIONAL

ERC-04 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	ERC20.sol: 1088	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.

addSniper()
removeSniper()
addLiquidityHolder()
removeLiquidityHolder()

Recommendation

We advise that the functions' visibility specifiers are set to external and the array-based argumentschange their data location from memory to calldata , optimizing the gas cost of the function.

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

Aug 2023