DeHacker

Code Security Assessment

Paraswap (new scope)

March10th, 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	Paraswap (new scope)
Platform	Ethereum
Website	https://www.paraswap.io/
Type	DEX
Language	Solidity

Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
PSP	PSP_BEP20.sol	21ccd5d4ac37f521779f76923febe402c1b71ca0adb80fad40bfd5f6dd7a928e

Findings

ID	Category	Severity	Status
PSP-01	Language Specific	Informational	Acknowledged
PSP-02	Centralization / Privilege	Major	Acknowledged
PSP-03	Logical Issue	Medium	Acknowledged

Informational

PSP-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	PSP_BEP20.sol: 1	Acknowledged

Description

The contract contains unlocked compiler versions. An unlocked compiler version in the contract's sourcecode permits the user to compile it at or above a particular version. This, in turn, leads to differences in thegenerated bytecode between compilations due to differing compiler version numbers. This can lead toambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult toidentify over a span of multiple compiler versions rather than a specific one,

Recommendation

It is general practice to alternatively lock the compiler at a specific version rather than allow a range ofcompiler versions to be utilized to avoid compiler-specific bugs and thus be able to detect emerging ones.We recommend locking the compiler at the lowest possible version that supports all the capabilitiesrequired by the codebase. This will ensure that the project utilizes a compiler version that has been in usefor the longest time and as such is less likely to contain yet-undiscovered bugs,

Major

PSP-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	PSP_BEP20.sol: 27, 35, 50	Acknowledged

Description

To bridge the gap in trust between the administrators need to express a sincere attitude regarding theconsiderations of the administrator team s anonymity.
The MINTER_ROLE has the responsibility to notify users about the following capabilities:
mint capped tokens to anyone through mint()
burn anyone' tokens through burn()
The DEFAULT_ADMIN_ROLE has the responsibility to notify users about the following capabilities:set minter' cap through
setMinterCap()
grant role to account through grantRole()
revoke a role from account through revokeRole()

Recommendation

We advise the client to carefully manage the privileged account's private keys to avoid any potential risksof being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to beimproved via a decentralized mechanism or via smart-contract-based accounts with enhanced securitypractices, e.g. Multisignature wallets.
Indicatively, here are some feasible suggestions that would also mitigate the potential risk at the differentlevels in terms of the 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.

Medium

PSP-03 | Volatile MINTER_ROLE Restriction

Category	Severity	Location	Status
Logical Issue	Medium	PSP_BEP20.sol: 37	Acknowledged

Description

The MINTER_ROLE should be restricted to mint up to the cap tokens. If the MINTER_ROLE mints afterburning another's token he/she can mint again. In this process, total is used to record the number ofmint , but as burn() reduce total , the actual cumulative number of mint can exceed cap .

Recommendation

We advise the client to recheck the logic.

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
- Eos
- Cosmos
- Substrate

TECH STACK

- Python
- Rust
- Solidity
- c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

March 2023