DeHacker

Code Security Assessment

Swash

April 7th, 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):

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire code base by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes.
- Add enough unit tests to cover the possible use cases.
- Provide more comments per each function for readability, especially contracts that are verified in public.
- Provide more transparency on privileged activities once the protocol is live.

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	Swash
Platform	Ethereum (ETH)
Website	https://swashapp.io/
Type	Others
Deployed contract	https://github.com/swashapp/token/tree/1df72e00ae7d942313a0646e17309ded86e93a86
Language	Solidity

Audit Summary

Delivery Date	April 7th, 2023
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	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	1	0	0
Informational	1	0	0	1	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
ERC	ERC677.sol	35da86421e2dba5a5803e51cee90d8ab422d69572bd76128887a97ec4aaf5fd8
IER	IERC677.sol	74bfc140c25fe68ec38de1d9ea868a8c5a472fc52fb4be7ae830268eb66b9411
IEC	IERC677Receiver.sol	244c3d302ca374df45f14ad97bb3f79c1f52a0573a58a74e599bf4570dab2ca0
SWA	SWASH.sol	83423b79f44e02e871c2ca97251c312c7120b35f815e80e7bac953a615f19bc2

Findings

ID	Title	Category	Severity	Status
ERC-02	Corner Case for Non-contract CallerCheck	Volatile Code	Informational	Acknowledged
SWA-01	Initial token distribution	Centralization /Privilege	Minor	Acknowledged

Informational

ERC-02 | Corner Case for Non-contract Caller Check

Category	Severity	Location	Status
Volatile Code	Informational	ERC677.sol: 49	Acknowledged

Description

Function isContract() cannot 100% guarantee the caller is a non-contract user, since EXTCODESIZEreturns 0 if it is called from the constructor of another contract. Please consider if this is a problem for the project.

Recommendation

We recommend checking msg.sender != tx.origin to confirm whether the user is a contract or not.

Alleviation

We have updated the recommendation. Also when _addr is a contract and transferAndCall() istriggered in the constructor of this contract, isContract() returns false, so transferAndCall() will not check if_to has the method onTokenTransfer().

Minor

SWA-01 | Initial token distribution

Category	Severity	Location	Status
Centralization / Privilege	Minor	SWASH.sol: 12	Acknowledged

Description

All of the SWASH tokens are sent to the contract deployer when deploying the contract. This could be acentralization risk as the deployer can distribute SWASH tokens without obtaining the consensus of the community.

Recommendation

We advise the client to be transparent regarding the initial token distribution process and the msg.senderaccount carefully to avoid any potential hack. In general, we strongly recommend that centralizedprivileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-basedaccounts with enhanced security practices, e.g., Multi-signature wallets.
Indicatively, here are some feasible solutions that would also mitigate the potential risk:
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.

Alleviation

[Swash Team]: We uses Gnosis Safe multi-sig contract with 6 owners and a threshold 3 to deploy thecontract and distribute SWASH tokens.

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/
- audit@dehacker.io

DeHacker

April 2023